FELIPE GOMES DE MELO D'ELIA
GIOVANNI CABRAL MORALES

AUTONOMOUS VEHICLE IMPLEMENTATION IN SCALE MODEL WITH ROS 2

São Paulo
2022

FELIPE GOMES DE MELO D'ELIA
GIOVANNI CABRAL MORALES

AUTONOMOUS VEHICLE IMPLEMENTATION IN SCALE MODEL WITH ROS 2

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro

Áreas de Concentração:

Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos

Departamento de Engenharia de Computação e Sistemas Digitais

Orientadores:

Profa. Larissa Driemeier

Prof. Jorge Risco

São Paulo
2022

RESUMO

O transporte rodoviário desempenha um papel de grande importância em nossa sociedade e no entanto impõe diversas questões para a vida humana, desde poluição até acidentes de trânsito. Acidentes de carro correspondem à maior causa de morte de jovens de 5 a 29 anos. Além disso, dados mostram que a principal causa de acidentes são erros humanos. Nesse contexto, o desenvolvimento de sistemas avançados de assistência ao condutor e além disso veículos autônomos surgem como uma possível solução para mitigar esse problema. Nosso trabalho apresenta uma revisão do estado da arte na pesquisa de veículos autônomos e oferece uma implementação de diferentes algoritmos utilizados na percepção e controle de veículos autônomos em um modelo virtual e em um protótipo físico. O projeto foi construído utilizando ROS 2, que possibilita o desenvolvimento de software modular e escalável. Por fim, uma discussão acerca dos resultados e uma conclusão sobre o projeto são apresentadas

Palavras-Chave – Veículos autônomos, Sistemas avançados de auxílio ao motorista, Robotic Operating System, Percepção, Controle.

ABSTRACT

Road transportation plays a major role in our society, yet imposes several problems for human life, such as pollution and traffic accidents. Car crashes correspond to the largest cause of death among young people from 5 up to 29 years. Beyond that, data shows that the vast majority of the vehicle accidents are caused by human error. In this context, the development of advanced driver-assistance systems and moreover autonomous vehicles arises as a possible solution to tackle this problem. This work presents an overview of the state of the art of research in the field of autonomous vehicles (AVs) and provides an implementation of different algorithms used in perception and control in the context of AVs in a virtual model and a physical prototype. The project is built using the software framework ROS 2, that enables the development of modular and scalable software. Finally, a discussion about the obtained results and a conclusion about the project are provided.

Keywords – Autonomous vehicles, Advanced driver-assistance systems, Robotic Operating System, Perception, Control

ACKNOWLEDGMENTS

To our main advisor Profa. Dra. Larissa Driemeier, for all the incredible support provided during the development of the project and for believing in our work since day one. To Prof. Dr. Jorge Risco, for the support, suggestions and for believing in this project. To the Prof. Rafael Moura for the discussions and advice.

To our friends Thalles Carneiro, Juliano Fonseca, Renato Yoshizaki and Lucas Guedes for the technical assistance offered during the development of the project and the fruitful discussions that helped us to move forward.

To the ThundeRatz Robotics Team for the experience and material support that made this project possible.

To the Polytechnic School of the University of São Paulo, for all the opportunities, knowledge and resources provided to us.

Felipe - To my family, my mother Fernanda, my brothers Lucas and Rafael, my godparents Renata and Luciano, my grandmother Alice and my cousins Bel and Bia, that supported me along the way and inspired me to always mode forward.

To all my friends from ThundeRatz, especially Isabella Bologna, Gustavo Oliveira, Felipe Bagni, Lucas Haug, Lucas Schneider, Jean Mello, Felipe Luna and Bernardo Coutinho, for the companionship during my time in the university and the amazing experiences that paved my way in the world of robotics.

To the friends from the university, Lucas Gaia, Douglas Bonfim, Glauco Eltink, Clara Cavalcanti and Hendrik Spreitzer, whose friendship made my journey lighter and kept me going forward.

Giovanni - To my parents Nadja and Regis, my brother Diego, my uncles Georgia and Adailton, my grandmother Nesita and my godmother Mazé for always incentivising and believing in me, providing help, love and support in every way they could.

To all the friends that i shared moments with in the last years, always being there to celebrate achievements and good moments or to comfort and cheer up each other in bad times. Thanks especially to Guilherme Boer, Caio Takamori, Caio de Souza and my mecatronic friends Luiz Fernando, Giulia Cardella and Lucas Borba.

To ThundeRatz, for the all the knowledge acquired, friends made and most of the best moments experienced in the university. Making my graduation full of excitement and challenges, always motivating me to keep improving and bringing the best of me.

The only way to discover the limits of the possible is to go beyond them into the impossible
- Arthur C. Clarke

LIST OF FIGURES

CONTENTS

1 INTRODUCTION

1.1 Motivation

Road transportation of people and goods is one of the main backbones of modern society. Despite its importance, road transport poses major negative externalities to our society, in the form of pollution, accidents and human casualties (13–15). The incidence of premature deaths resulting from pollution inhalation from traffic congestion in the US is estimated to be over 20 billion USB in 2010 (16). Road accidents are the leading cause of deaths of young people in the age between 5 and 29 years and the 8^{th} leading cause of deaths across all ages, surpassing diseases like HIV/AIDS and tuberculosis (17, 18). According to a study conducted by the US agency National Highway Traffic Safety Administration (NHTSA) (19), more than 90% of car accidents in the US are caused by human error.

In pursue of a more precise driving system than a human-based one, autonomous vehicles (AVs) research has shown positive results in reducing traffic in urban areas (20) and increase energy usage efficiency (21), in addition to its capability to free people from the mental and physical burden of long travels. In this context, the present work aims to provide an implementation of autonomous vehicle in a scale model with the software framework ROS 2, leveraging its abstraction capabilities in order to provide a solution that could be applied to a real scale vehicle.

1.2 Autonomous vehicles

The idea of autonomous vehicles (AVs) dates back the 1920s (5, 22), with diffusion of the concept in television as early as 1958 (23). It was only in the 1980s that the idea came closer to reality, with major accomplishments in the scope of the european PROMETHEUS research project (24, 25), with a major demonstration in 1994, when the project's AV developed in the Universität der Bundeswehr München, VaMP (acronym

from german Versuchsfahrzeug für autonome Mobilität und Rechnersehen), drove 1,600 km, of which 95% autonomously (25,26). In the same period, Carnegie Mellon University Navigation Lab (CMU NAVLAB) made remarkable advance in the area and in 1995 made a similar demonstration with a 5,000 km drive across the US, of which 98% autonomously (27).

The next important historical marks of AVs development are the competitions held by the US agency DARPA (Defense Advanced Research Projects Agency), that in 2004 and 2005 organized the DARPA Grand Challenges (28) and in 2007 organized the DARPA Urban Challenge (29). In the 2004 and 2005 DARPA Grand Challenge, the objective was to navigate through a 150-mile off-road course as fast as possible without any human intervention (30), in contrast to previous works that had minimal (but no zero) interventions. No team was able to complete the course in the first edition in 2004 and in the second edition held in 2005, 5 of 23 teams reached the finish line (30). In the 2007 DARPA Urban Challenge, the objective was to navigate inside a simulated urban environment and 6 of 11 teams succeeded, as major demonstration that AVs could inside cities (31).

To classify different degrees of autonomy, the US National Highway Traffic Safety Administration (NHTSA) created a 4-level taxonomic classification in 2013 (32) and the Society of Automotive Engineers International (SAE) created a 5-level taxonomic classification in 2014, later revised with an extra classification corresponding to level 0 as complete absence of autonomy (33) as detailed in Figure 1. The electric cars company Tesla claims that its model S reached level 2.5 of autonomy, while new Audi A-8 reached level 3, turning Audi the first automaker to offer a vehicle with such high level of autonomy (34).

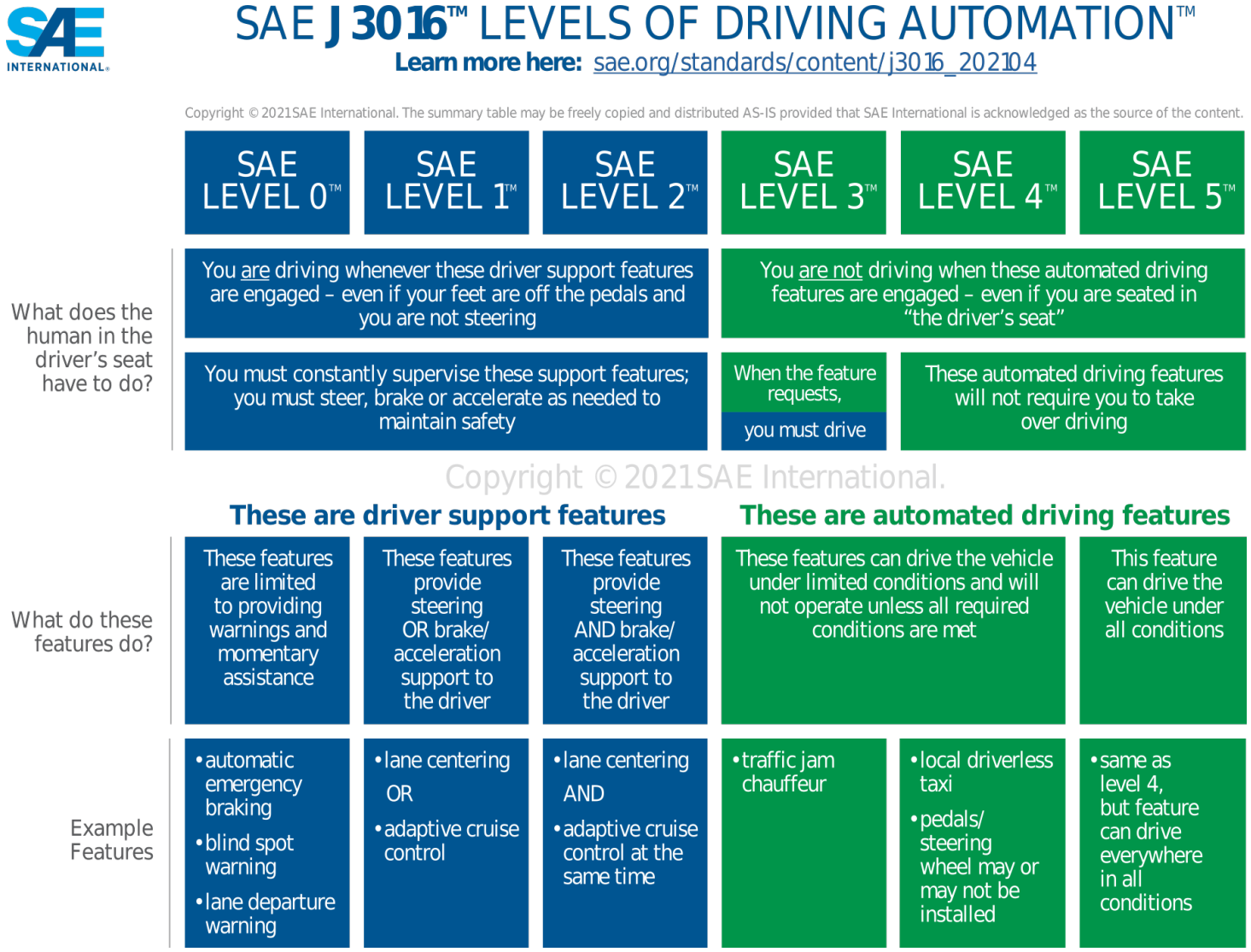

Figure 1: SAE J 3016 levels of driving automation. Source (2)

1.3 Robot Operating System (ROS)

The Robot Operating System (ROS) is a project developed by Willow Garage Organization, as an open source initiative to enable code reuse among different robotics projects. It's based in a distributed and centralized *peer-to-peer* structure, where several processes, called "nodes", communicate with each other with well defined interfaces and with the intermediation of a master node called `rosmaster` (35).

The second generation of ROS, ROS 2, was redesigned from scratch in order to solve some core issues that ROS was not designed to solve, such as security and real-time capabilities and support for large scale embedded systems. ROS 2 is based on the Data Distribution Service (DDS), an open source standard already used in critical infrastructure such as military, spacecraft and financial systems (36). This change leverages an already existing and adopted interprocess communication standard and removes the burden of maintaining such standard from ROS 2 core development team (37).

It is possible to highlight some of ROS 2 main features:

Client libraries a common client library implemented in C language `rclc` is introduced, in such a way that client libraries for different languages (e.g. C++, Python, Rust) can be extended from C low-level implementation (3, 37). In contrast, ROS 1 client libraries are implemented from scratch, which lead to inconsistent behaviors and naming conventions across different client libraries and the burden of solving a same bug in multiple libraries (3, 38).

Middleware interface ROS 2 defines a common middleware interface called ROS middleware interface (RMW) that can be extended by any DDS implementation in such a way that the client libraries are completely independent from the underlying transport method (39).

Real-time By adopting DDS as transport middleware, ROS 2 aims to provide real-time performance to allow its adoption by safety-critical systems (40).

Security Another set of desired features from the DDS standard are security capabilities, such as authentication, access control and cryptography (41, 42).

An illustration of ROS 2 structure can be seen in Figure 2.

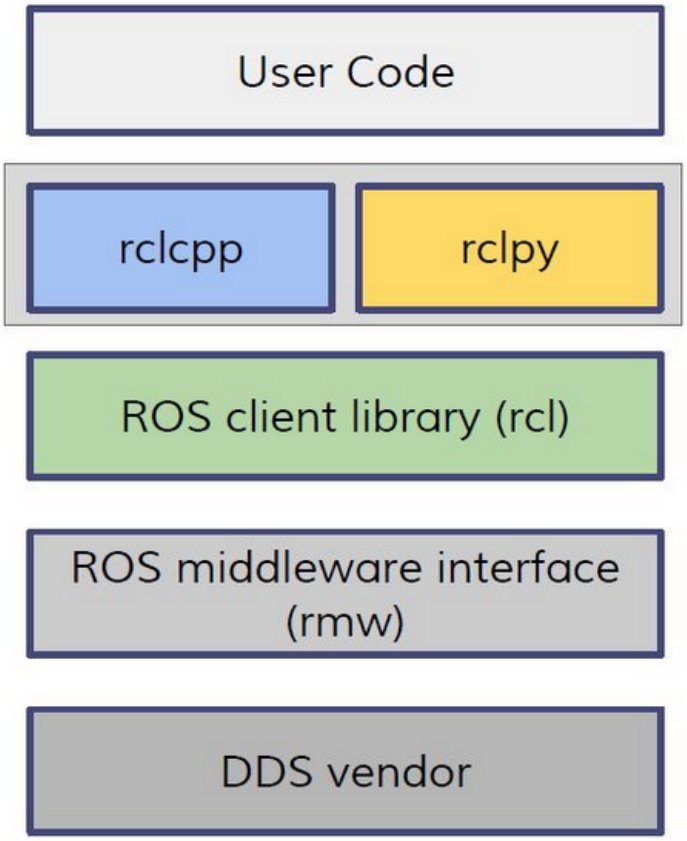

Figure 2: Illustration of ROS 2 core stack. Source (3)

1.4 Data Distribution Service (DDS)

The Data Distribution Service (DDS) is a standard maintained by the Object Management Group (OMG), whose purpose is to provide a common application-level interface that clearly defines the data-distribution service. The standard seeks to unify existing implementations by enumerating and providing formal definitions for the Quality of Service (QoS) settings that configure the behaviour of the service (36). DDS follows a publish-subscribe pattern and provides a mechanism for dynamic discovery of both publishers and subscribers, in such a way that there is no need for a centralized entity to manage the existing processes as the `rosmaster` in ROS 1. The decentralized nature of DDS is shown in Figure 3.

1.4.1 Supported DDS providers

By design, ROS 2 is build in such a way that is possible to use different DDS implementations provided by several vendors. The eighth release of ROS 2 Humble Hawksbill, released in May 2022 (43), provides support out-of-the-box for 4 different DDS implemen-

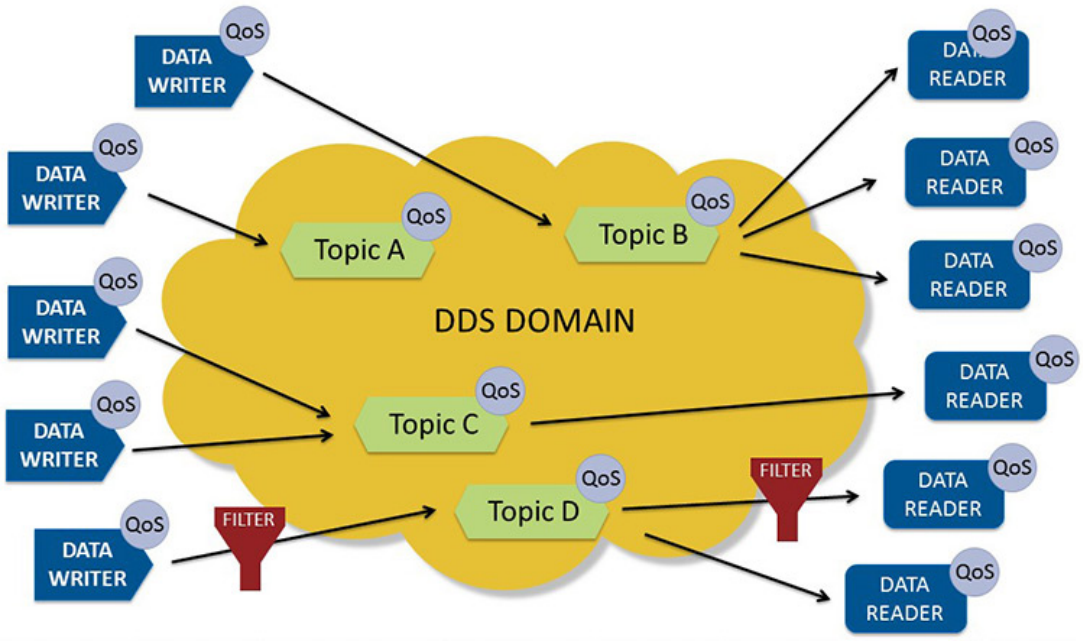

Figure 3: Decentralized structure of DDS standard. Source (4)

tations (44, 45).

Fast DDS default DDS implementation utilized by ROS 2 Humble, supported since ROS 2 first release (45). It is provided by eProsima with the open-source license Apache.

Cyclone DDS provided by the Eclipse Foundation with the open-source license Eclipse Public License v2.0. It first received supported in the ROS 2 fourth release Dashing and is the successor of the OpenSplice Community Edition, whose support for ROS 2 was dropped in the sixth release Foxy (45, 46).

Connext DDS provided by RTI with commercial and research licenses, with full support since ROS 2 first release (45).

GurumDDS provided by GurumNetworks with commercial licenses, with community provided support since ROS 2 sixth release Foxy (45).

From the first ROS 2 release, Ardent, until the sixth release, Foxy, the default DDS implementation was Fast DDS. In order to chose the default DDS implementation for the seventh release, Galactic, the Technical Steering Committee (TSC) performed a benchmark among the supported DDS implementations with Tier 1 support and permissive open source licenses (47, 48) and chose Cyclone DDS as default provider for the G-release (49). In 2021, the benchmark was performed again (50) and the TSC chose Fast DDS as default provider for ROS 2 eighth release Humble (43). For the releases after Humble, the

TSC decided to only change the default DDS provider in non Long Time Support (LTS) releases, in order to ensure more stability to LTS distributions (51).

1.5 ROS 2 benchmarks

Several benchmarks were performed to asses the performance of ROS 2 as middleware in autonomous systems (52–56). One of the firsts works in the subject, Maruyama et al. (2016) (52) presents a comparison between ROS 1 and different quality of service (QoS) settings for ROS 2, which showed different DDS vendors with similar performance as ROS 1 `nodes` and `nodelets`. In the same year, Dabrowski et al. (2016) (53) presented a similar analysis focusing in lossy networks and characterized the degradation behaviour of both **Connext DDS** and **Fast DDS** upon increased packet loss. Both perform in a similar way and the authors point that the chosen QoS settings display the greatest impact in the performance of the system. A common conclusion of both works is the positive gain of adopting DDS as transport middleware, as it stands as well established transport method, but highlight the needed effort to understand its QoS options as they can heavily influence the systems performance.

Pemmaiah et al. (2018) (54) provides an in-depth analysis of the process of building a test tool to properly assess the system's performance. It presents relevant parameters that may affect the measurement, such as chosen QoS policy, different transport mechanisms and buffer sizes. It also discuss the trade-off between application's determinism and high-performance.

Fernandez et al. (2020) (57) explores the impact of the security features available in the DDS standard in the performance of data exchange. It shows that it is possible to use cryptography algorithms such as Rivest-Shamir-Adleman (RSA) 2048 bit and Elliptic Curve Cryptography (ECC) 256 bit to protect encrypted connections in the context of warfare systems.

Puck et al. (2021) (58) evaluates the real-time capabilities of ROS 2 with the *ros2_tracing* (56) tool. It employs a set of industrial PCs (IPCs) running a real-time Linux kernel (i.e. Preempt-RT) and synchronized via the precision timing protocol (PTP). It shows that of-the-box settings of ROS 2 are not real-time ready, but upon some modifications it is possible to achieve real-time safe performance for low control frequencies (under 1kHz).

Kronauer et al. (2021) (55) provides an analysis of multi-node communication chains

and how the number of nodes can affect the performance of data exchange using **Fast-DDS**, **CycloneDDS** and **Connext**. It shows that latency is highly dependent on hardware and chosen QoS settings, with no DDS provider displaying best performance in all cases.

2 STATE OF THE ART

The complex task of autonomous drive is usually split in a 3 steps procedure: perception, planning and control (5, 8, 26, 59–62) as illustrated by Figure 4.

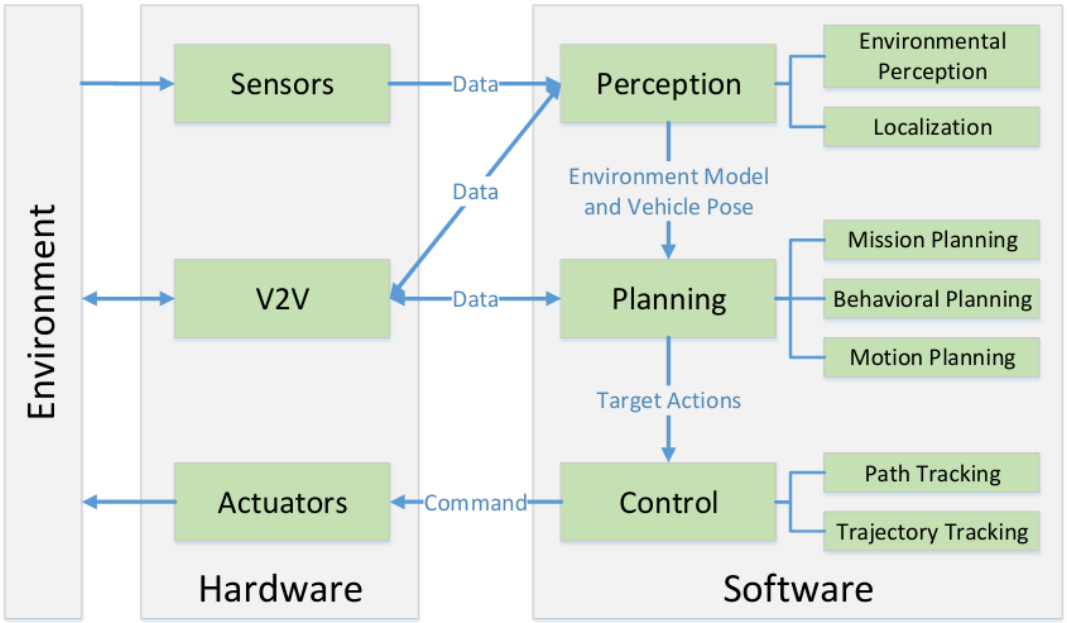

Figure 4: Typical autonomous vehicle system overview proposed by (5)
V2V refers to Vehicle-to-Vehicle communication

Perception refers to the ability of an autonomous system to collect information and extract relevant knowledge from the environment (5).

Planning refers on how a robot takes decisions in order to perform a high level mission (5), such as reaching a desired location or performing a complex task. It is usually divided in global (or path) planning, behaviour planning and local (or motion) planning (5, 8, 63–65). Global planning refers to high level planning, such as the route to reach a desired location, behaviour planning refers to decisions to ensure the vehicle interacts with the environment in a safe manner (i.e. following traffic laws and streets signals), and local planning refers to immediate decisions to accomplish the current global goal, such as when to change lanes (5, 65).

Control refers to the ability to transpose desired actions generated by planning algorithms to the real world. According to (5), "refers to the ability to execute the planned actions that have been generated by higher level processes". It is accomplished with techniques such as classic control approaches as PID controllers, modern control approaches such as Model Predictive Control (MPC) or machine learn based approaches such as Reinforced Learning (RL) (8).

An example of system implemented following this structure is Autoware.Auto Autonomous valet Parking demo, whose architecture is shown in Figure 5.

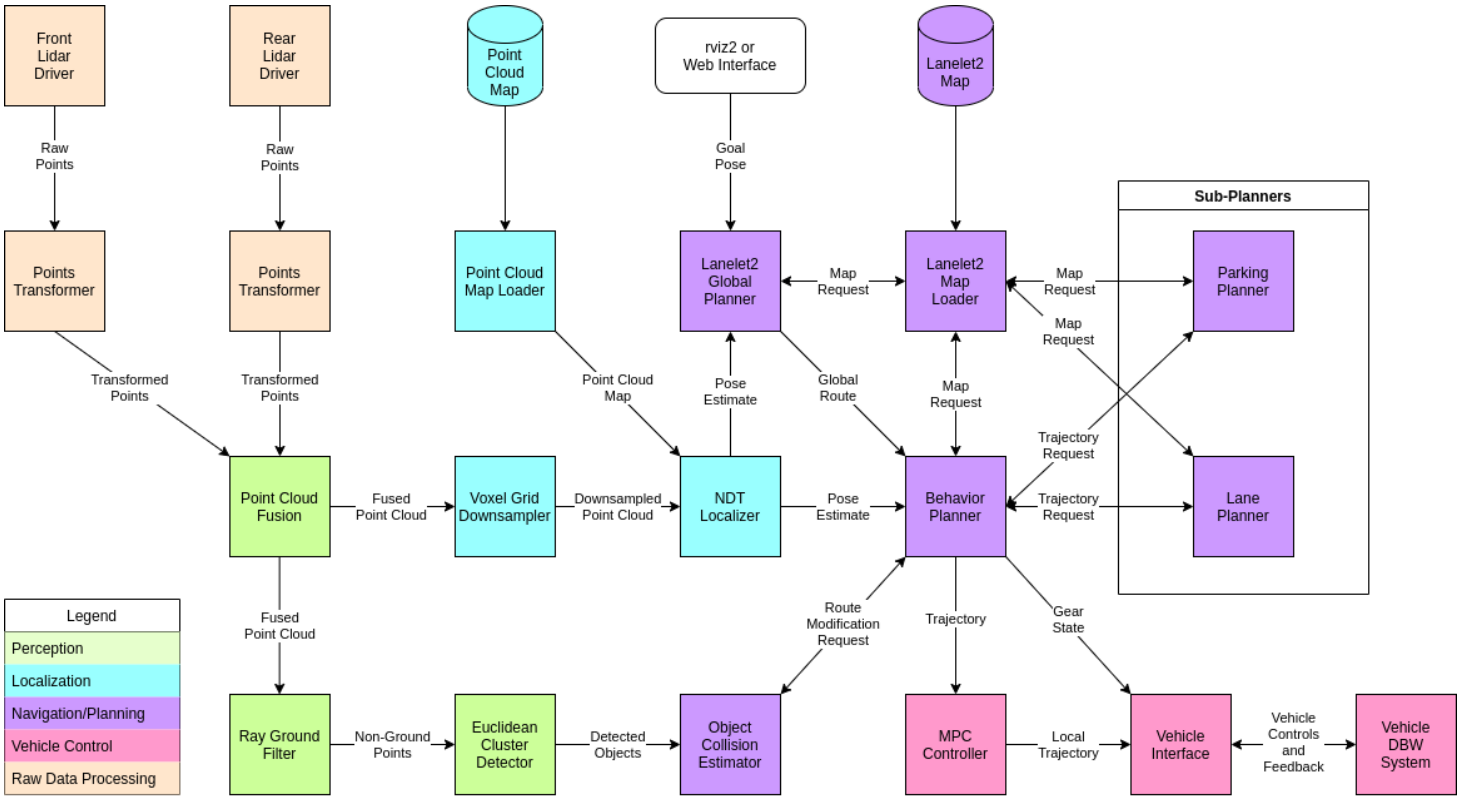

Figure 5: Autoware.Auto Autonomous Valet Parking demo software architecture Available at (6)

2.1 Perception

Perception can be divided primarily in two branches: localization and object detection.

2.1.1 Localization

Many localization approaches heavily rely on LiDAR sensors, described as a "game changer" in the field (66), as it is able to create a dynamic, three-dimensional map of the

environment by sending millions of light pulses per second with its rotating axis. However the data returned by those sensors are not perfect, with problems as point sparsity, missing points, and unorganized patterns (5).

Simultaneous Localization and Mapping (SLAM) algorithms use probabilistic tools to synthesize information from LiDAR scans to build and update the map of the unknown environment, together with the tracking of the agent's location and orientation.

Analyzing different SLAM approaches, GraphSLAM is currently the most accurate one (67), outperforming other options, like EKF-SLAM, by up to 15 times when fusing data from different vehicle sensors (e.g. wheel encoder, IMU) (68), although it has a higher memory cost.

Due to the high cost of LiDAR sensor, novel SLAM approaches aim to obtain the same result as LiDAR-based solutions but using stereo (also called RGB-D) cameras. As these approaches rely on visual information from a set of images, they are also referred as VSLAM algorithms (66), with main exponents as ORB-SLAM3 (69), OpenVSLAM (70) and RTAB-Map (71).

2.1.2 Object detection

Being able to detect objects is an essential task in some of autonomous driving objectives. Vehicles, pedestrians and cyclists are key objects for accident prevention and driver assistance. Some of the principal challenges are related to object occlusion and prediction (72). Likewise, lane line marks and road surfaces are crucial to a reliable urban and highway traffic, whose detection issues are mainly challenging light conditions, affecting the road unevenly, and missing land marks (73).

In recent years, deep learning approaches for detection have shown superior performance and reliability compared to conventional learning or feature based methods. However, most of these implementations result in a higher processing time. Some other approaches focus on achieving real time performance, e.g. Darknet You Only Look Once (YOLO) (74–77) and Single Shot Detection (SSD), enabling applications that rely on dynamic and constant data processing to use object detection as its main perception tool.

2.2 Planning

2.2.1 Global planning

Classical approaches to global planning are based in graph search algorithms over a directed graph network that represents roads connectivity, such as Dijkstra's (78) and A* (79), with good performance in small neighborhoods but poor scalability (5). More recent approaches rely on optimization methods (8) and reinforced learning (80–82) with advantage of smaller querying times in comparison to classic path searching algorithms (83).

2.2.2 Behaviour planning

To perform tasks such as react to sudden environment changes and interact with other agents in a safe manner, many AVs at the DARPA Urban Challenge implemented Finite State Machines (FSM) (29, 84–86). More recent approaches use Behavior Trees (BT) instead of FSMs (64), that show better reactivity, modularity and scalability in relation to FSMs (87). Some novel approaches perform optimization across a trajectory space generated by the local planner and select the one with lowest cost in relation to a given cost function. This cost function may take into account values such as proximity to obstacles, control effort, acceleration and jerk, in order to maximize safety (by minimizing distance to obstacles) and comfort (by minimizing acceleration and jerk) (8).

2.2.3 Local planning

The main objective of local planning is to plan the motion of the car in a finite fixed time horizon in order to make progress along the current route generated by the global planner (8). Motion planners can be evaluated according to their *computational efficiency* and *completeness*. Computational efficiency refers to the process run time and its scalability based on the dimensionality of the configuration space. The algorithm is considered *complete* if it ceases its execution in finite time either returning a solution or indicating that no solution exists (88).

Classical local planners can be divided as *combinatorial* or *sampling-based* planners. Some examples of *complete* combinatorial planners are visibility graphs (provides shortest path solution) and Voronoi-diagrams (provides highest clearance solution) (5, 65). The main drawback of these methods is their computational burden, specially

in highly complex environments, which lead to sampling-based methods development (89–91). Sampling-based methods provide *probabilistic completeness*, which means that if sufficient time to check an infinite number of samples is given, the probability of finding a solution (if one exists) converges to 1. Some prominent works in this area are Probabilistic RoadMaps (PRM) (91, 92) and Rapid-exploring Random Trees (RRT) (89, 93).

Other optimization based methods used in local planning are the Dynamic Window Approach (DWA) (94) and Time Elastic Band (TEB) (95), both utilized by the ROS Navigation Stack (63, 64).

2.3 Control

The control competency of an autonomous system refers to the process of converting intentions, generated by higher level planners, into actions (5). It's the lowest abstraction level in the software stack, since its outputs are directly transformed into hardware commands.

One traditional form of classic control is the Proportional-Integrative-Derivative Controller (PID), tunned to path following, in order to minimize the lateral, longitudinal, heading error (7, 96), as shown in Figure 6.

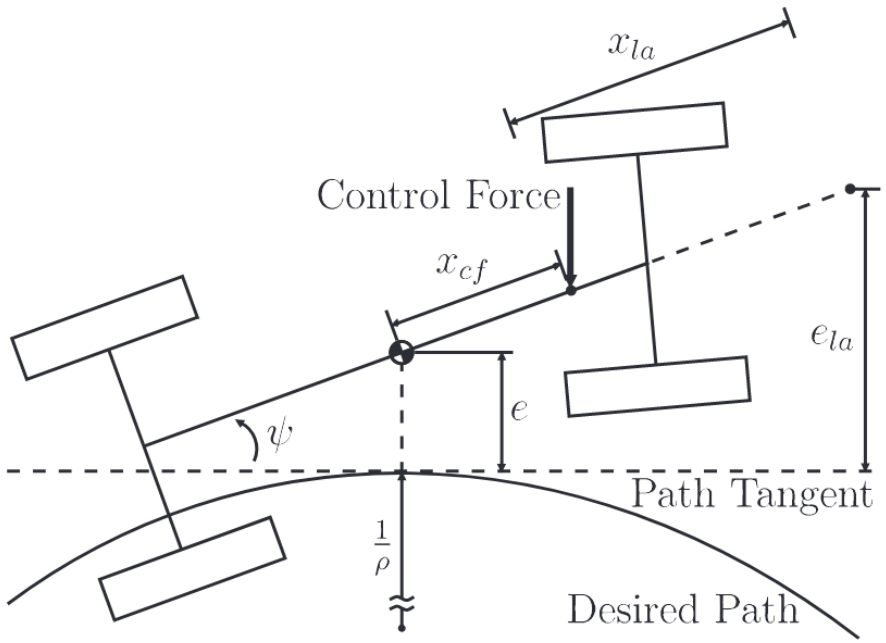

Figure 6: Definition of longitudinal error e and heading angle error ψ from (7). The path local curvature is described by ρ

The mathematical formulation of the classic PID controller follows the time-domain

equation

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

where $u(t)$ is the control effort, $e(t)$ is the control error and K_p, K_i and K_d are the proportional, integral and derivative controller gains, respectively.

In order the tackle the classic PID controller limitations, such as delayed response to errors and coupled responses to disturbances, modelling error and measurement noise, another degree of freedom may be added to the control system by including a feedforward term to the controller, as shown in Figure 7.

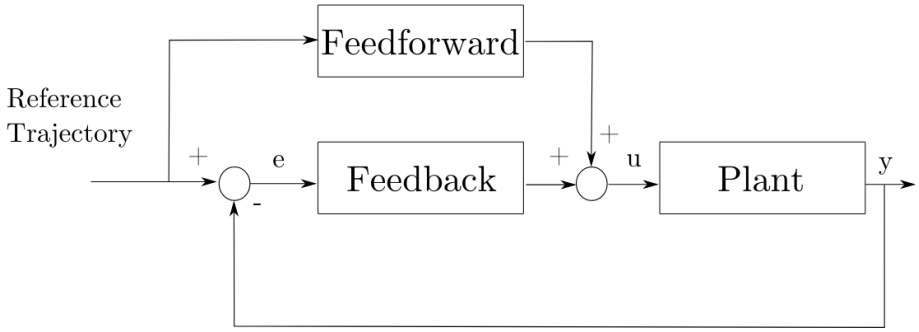

Figure 7: Two degrees of freedom PID controller as illustrated by (5)

2.4 End-to-End

As a complementary approach to the classic perception-planning-control pipeline, some works try to achieve *end-to-end* solutions for autonomous driving with use of techniques such as optimization, deep learning or reinforced learning (8) as shown in Figure 8.

As displayed by Betz et.al. (8), most works up to 2019 used an Optimal Control Policy, implementing it with methods such as Fuzzy Logic and Evolutionary Algorithms. Most recently, Deep Learning and Reinforcement Learning (RL) dominated the field, especially when paired with Vision based planning models.

Model based RL is currently the state-of-art, outperforming non-model based methods while displaying good generalizations on unknown track (97, 98).

Some recent approaches to AVs control rely on Model Predictive Control (MPC) to perform path following, a basic structure of it is illustrated in Figure 9. The MPC has

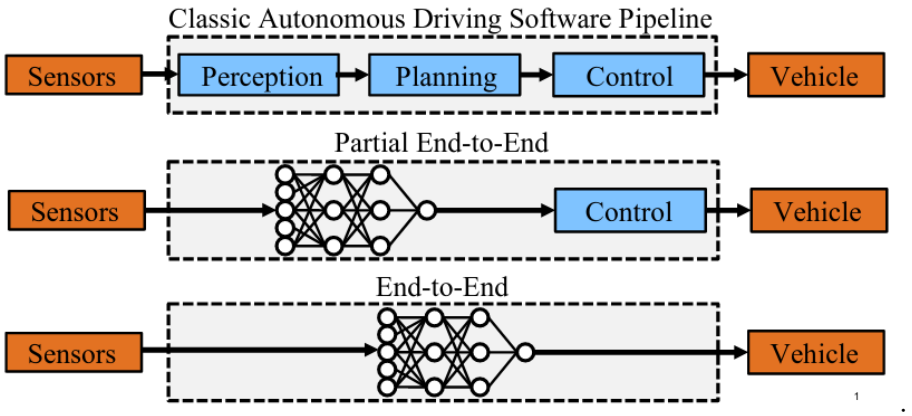

Figure 8: Comparison between classic autonomous driving solutions to partial or complete end-to-end software pipelines as illustrated by (8)

been developed to integrate the performance of optimal control and robustness of robust control (5). It solves a *Finite Time Optimal Control Problem* (FTOCP) to compute an optimal sequence of inputs $\{u_t^*, \ldots, u_{t+N-1}^*\}$ and states $\{x_t^*, \ldots, x_{t+N}^*\}$ over a fixed horizon N in order to minimize a certain cost function (8). The FTOCP is solved at each time step, so the controller take into account the feedback from the environment as the action is performed and can take proactive actions as the state of the system is evaluated in future time steps, enabling the system to adapt to distinct situations. Solving this optimization problem may be a computational challenge and is addressed through stochastic (99) or learning based (100) methods.

Some of MPC attractive features are its ability to handle complicated process models with input constraints and non-linearities (101) and simplicity of designing a multi variable feedback controller (5).

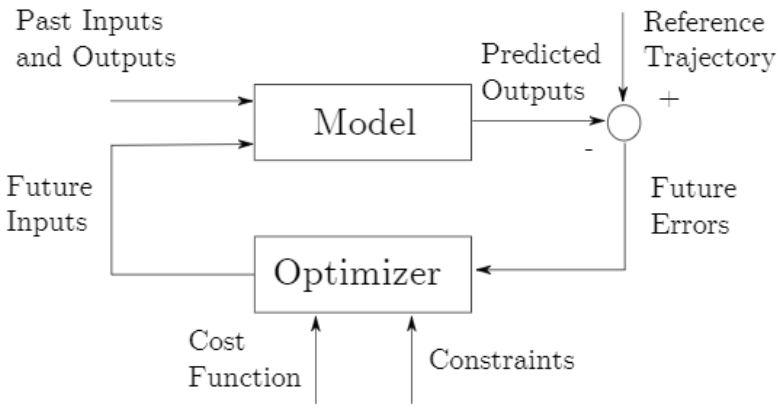

Figure 9: Basic Structure of Model Predictive Control as shown in (5)

3 METHODOLOGY

3.1 Scope of the project

The main focus of this work is the project and development of an autonomous driving system to be implemented in a simulated environment and then in a scale prototype. The system should be able to perform maneuvers in a controlled and static environment (e.g. valet parking) and should be able to provide active assistance to the driver in order to prevent collisions. In the scale defined by SAE J3016 (shown in Figure 1), the system should fulfill the requirements for level 2 autonomous driving.

The system was first developed inside the CARLA simulator (12) using a virtual model of a Tesla Model 3, with a custom set of sensors defined to best suit our autonomous driving system. The definition of the dynamic properties of the model are beyond the scope of this work, so an existing one is used in the project.

In the next phase of the project, the system was implemented in a 10:1 scale model using a similar set of sensors as the simulated model. The model was built from parts commercially available and some needed mechanical adaptations were made in order to embed the selected sensors and control hardware in the model.

3.2 Definitions

From the ISO 26262-1 (102), the following definitions are used in the text:

Fault "Abnormal condition that can cause an *element* or an *item* to *fail*"

Failure "Termination of an intended behaviour of an *element* or an *item* due to a *fault* manifestation"

Fault Tolerance "Ability to deliver a specified functionality in the presence of one or more specified *faults*"

Harm "Physical injury or damage to the health of persons"

Hazard "Potential source of *harm* caused by *malfunctioning behaviour* of the *item*"

Risk "Combination of the probability of occurrence of *harm* and the *severity* of that *harm*"

Functional safety "Absence of *unreasonable risk* due to *hazards* caused by *malfunctioning behaviour* of *electrical or electronic systems*"

3.3 Methodology of the project

In accordance with the presented state of the art, the project is divided in 3 subsystems: sensing, planning and control. Each subsystem is composed by one or more processes that exchange information with each other in real time. In order to achieve consistent fault tolerance, a redundant system is desirable when possible, so the vehicle can still work if one of its elements fail.

The project follows a hybrid approach mixing simulation and real world tests. The development starts inside a fully virtual environment and, once the system is capable of performing its tasks in the simulation, it is then tested in the real scale model. The virtual environment is still used as auxiliary tools along the entire project. This approach is illustrated in Figure 10.

The performance of the system was evaluated in a set of repeatable tasks in simple scenarios. The Carla simulator provides a common set of evaluation scenarios, in such a way that is possible to compare the results of our system in comparison with different benchmarks.

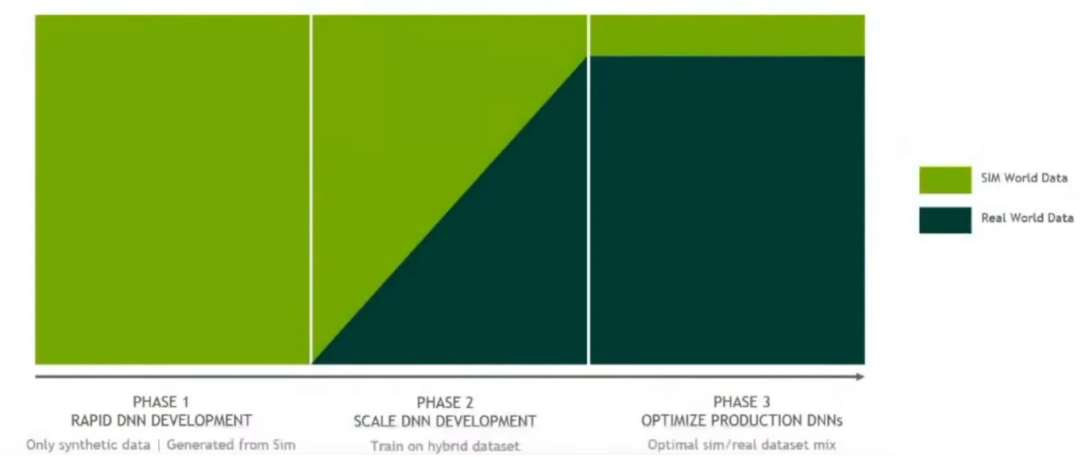

Figure 10: Simulation and real world approach as proposed by Nvidia, taken from (9)

3.4 Software structure

The work follows the structure of a ROS 2 project, which is composed by a series of packages, each one with a single responsibility. The project follows the organization standard defined by REP-144 (103) and the best practices guidelines from the ROS community (104).

The minimal unit of a ROS 2 package is called node, that communicate with each other using ROS 2 messages (37). The structure of the communication follows the dependency inversion principle, where high-level and low-level processes depend on an abstract interface between them instead of the implementation itself (105). In our project, the abstract interfaces are described by the used ROS 2 messages.

4 REQUIREMENTS

4.1 Functional requirements

The system should fulfill the requirements for level 2 of autonomous driving (i.e. steering and brake/acceleration assistance) in SAE J3016 standard (33), as depicted in Figure 1. The desired features are:

Maneuver assistance The system should provide assistance to the car driver to perform simple maneuvers in controlled environments (i.e. parking lot).

Navigation with landmarks The system be able to localize itself in respect to miniaturized traffic cones. The traffic cones should follow a 10:1 scale in respect to the Brazilian standard NBR 15071 (106).

Collision avoidance The system should prevent commands that may cause a collision with the landmarks or any other object in the surroundings.

4.2 Non functional requirements

Modularity As described in the previous chapter, AV control is a highly complex task. In order to tackle this complexity, a modular architecture design is desirable. From (107) "A complex system can be managed by dividing it up into smaller pieces and looking at each one separately. When the complexity of one of the elements crosses a certain threshold, that complexity can be isolated by defining a separate abstraction that has a simple interface. The abstraction hides the complexity of the element; the interface indicates how the element interacts with the larger system"

5 DEVELOPMENT

5.1 Virtual model

In accordance with state of the art approaches shown in, a virtual model was built in order to validate the developed software in early stages of development and ease the test of high level features. The model was build inside the CARLA simulator and is based upon the already available Tesla Model 3(108) virtual model, with a custom set of sensors in order to match the sensors used in the physical scale model. The main sensors used in the vehicle are a set of 4 stereo cameras, positioned in all sides of the car as illustrated in Figure 11.

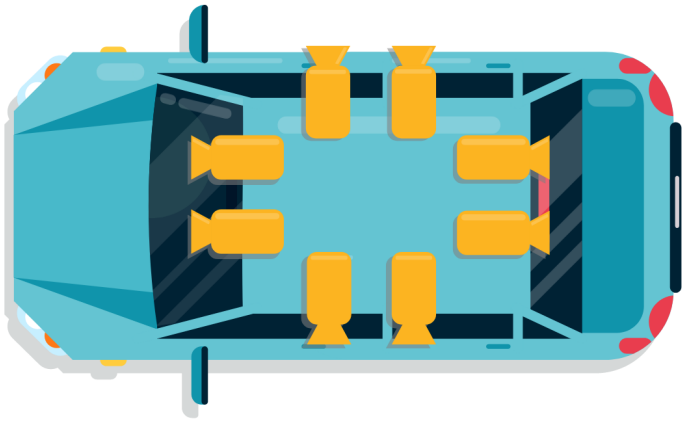

Figure 11: Illustration of cameras disposition in the virtual CARLA model

The interface with the model is done through the CARLA ROS bridge package (109) with custom ROS topics defined by the CARLA interface. At each simulation tick, the control system receives data generated by the sensors, computes the control action and then sends it to the simulator.

5.2 Hardware prototype

In addition to the virtual model, a physical 1:10 scale prototype was also built in order to validate the system's performance in real world conditions, shown in Figure 12. It is divided in 3 sub-systems, mechanical power train in the bottom level, electrical power train in the middle level and finally sensing and processing in the top level, as shown in Figure 13.

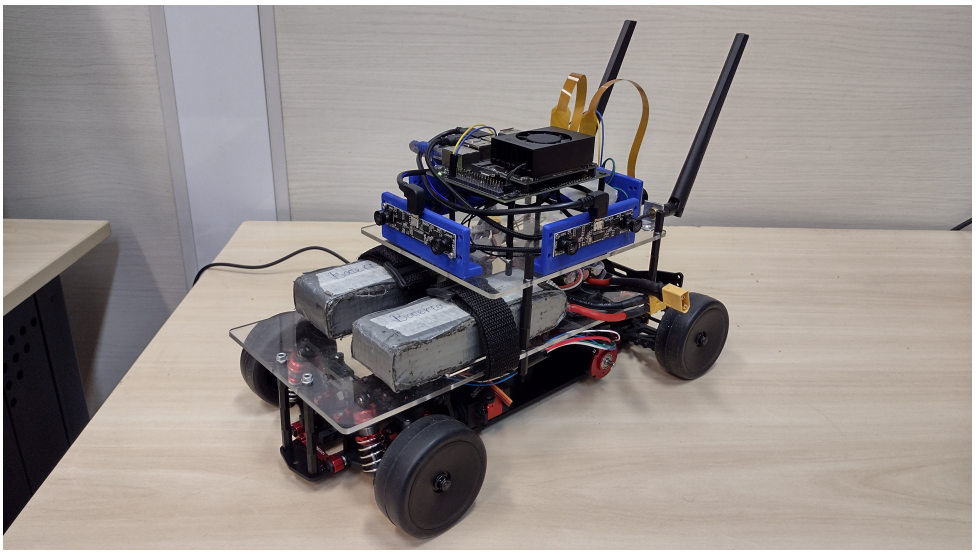

Figure 12: Physical prototype built by the authors

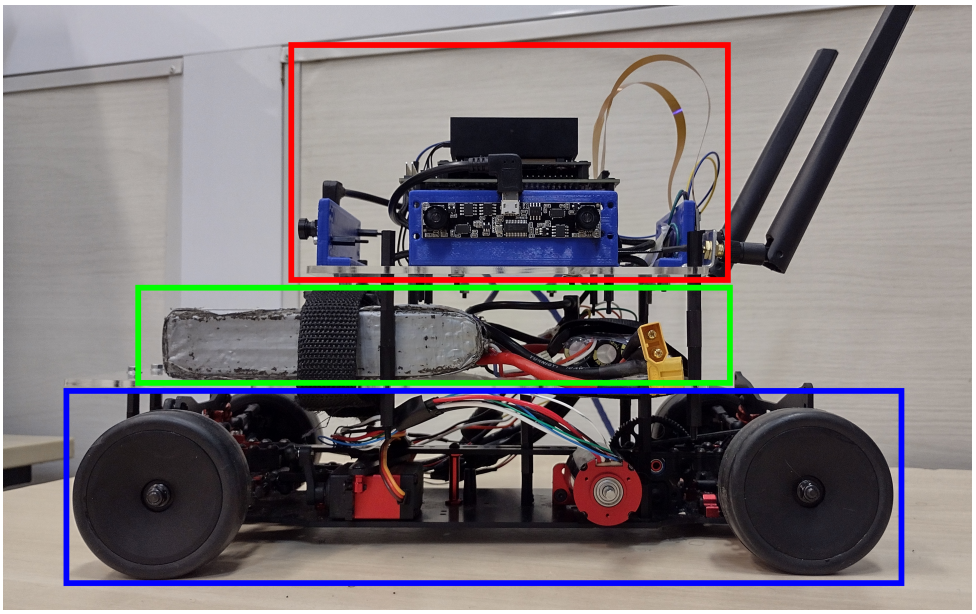

Figure 13: **Red** sensing and processing sub-system, **green** electrical power train and **blue** mechanical power train

The mechanical power train is composed by a Maxon EC-4 pole 30mm 24V brushless motor (110) connected to a gear transmission system with a differential gear train to

the rear wheels, allowing then to rotate in different speeds if needed. The mechanical structure was built over the R2-G SNRC 10:1 electric touring car kit, shown in Figure 14. The original kit has 4 wheel traction, but as only the rear wheels have a differential transmission, it was decided to disconnect the transmission to the front wheels in order to prevent slip and thus improve controllability.

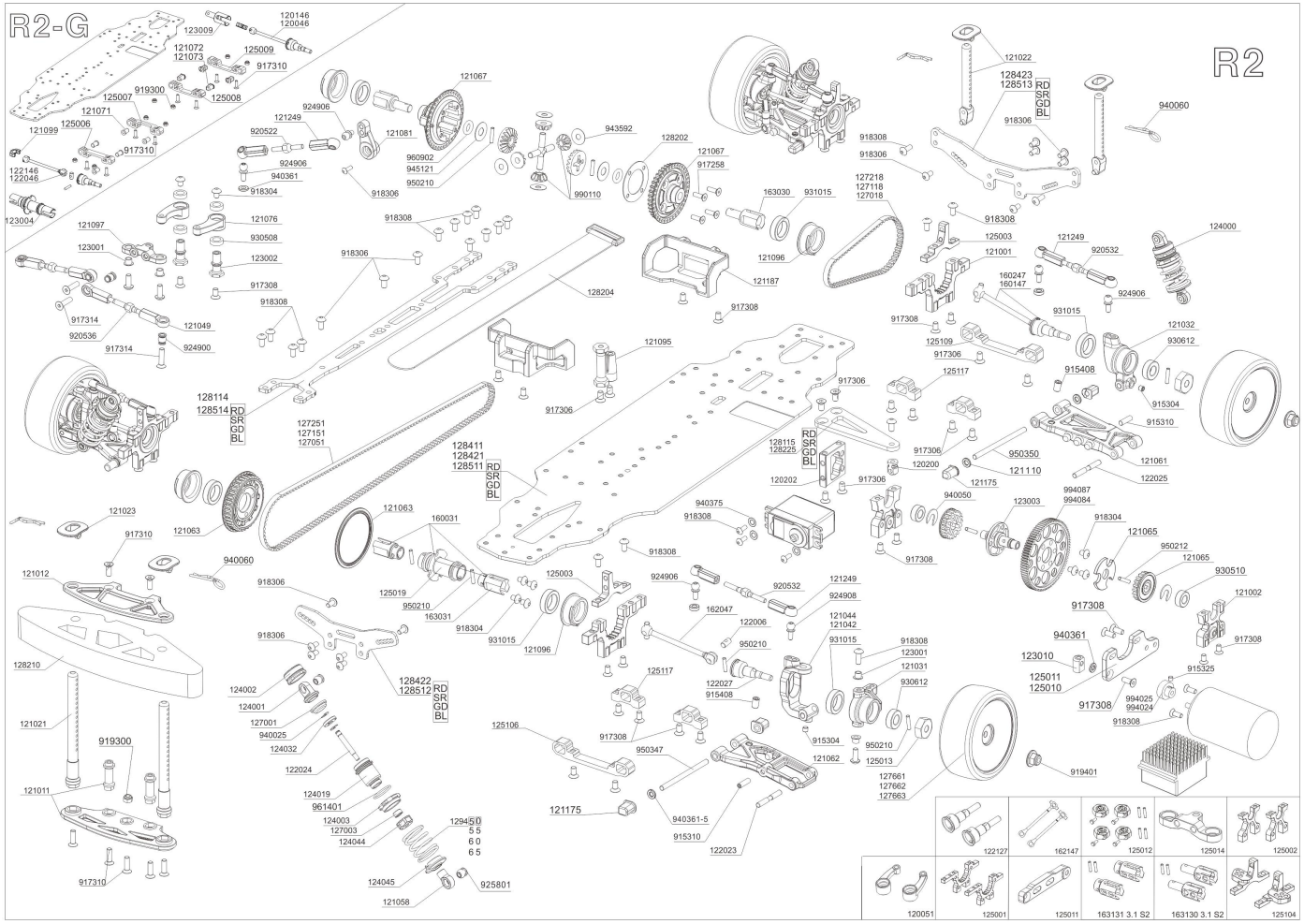

Figure 14: Exploded view of the R2-G RC car chassis provided by the manufacturer. Available at http://sn-rc.com/en/mcart.jsp

The electrical power train is composed by a VESC-based electronic speed controller and 2 3-cell 2000mAh LiPo batteries. The VESC (Vedder electronic speed controller) project is an open source project created by Benjamin Vedder with focus to provide a free and open source solution to control electrical motors in the context of robots, e-bikes, electric skateboards and others (111). One of the batteries powers the VESC and the other is responsible to power the sensing and processing sub-system.

The sensing and processing sub-system is composed by one Jetson Xavier NX as main processing unit, 3 ELP 1MP 120FOV USB stereo cameras and 1 IMX218-83 CSI stereo camera. This camera setup was chosen due to the limitation of number of USB ports in the Jetson Board, in such a way that one of the cameras takes advantage of the Camera Serial Interface (CSI) hardware available in the board. The Jetson Xavier NX platform provides a 384-core GPU and a 6-core CPU, capable of up to 21 Tera Operations per Second, and so it is an ideal embedded platform for image processing.

In addition to the scale car, a set of miniaturized traffic cones was manufactured with a fused material deposition (FDM) 3D printer, the results are displayed in Figure 15. In accordance with a 10:1 scaled version of the NBR 15071, the traffic cones have a square base of 50mm and height of 70mm.

5.3 Perception

5.3.1 Camera calibration

The images provided by real cameras are subjected by a set of distortions due to imperfections in the manufacturing process. An in-depth analysis of camera image distortions and their causes may be found in (112) and (113). In the present work two main sources of distortion are considered, radial and tangential.

The radial distortion corresponds to the displacement of pixels near the edges of the image and it is responsible for the barrel and the pincushion distortion. The distortion is zero in the center of the image and it increases with the radial distance r. This relation may be described by the Taylor polynomial in respect to the radial distance r. For typical web cameras, a third order polynomial is sufficient to model this distortion (114)

Figure 15: Traffic cones used in the project

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

The tangential distortion occurs due to a misalignment between the camera lens and its optical sensor, in such a way that the image is skewed and its center is displaced. It may be modeled by the relation (115)

$$x_{distorted} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

Therefore, a camera distortion may be characterized by the set of 5 coefficients $[k_1\ k_2\ p_1\ p_2\ k_3]$. The calibration of all cameras were performed using a 6x8 chessboard pattern printed in a A4 paper sheet and fixed in a solid frame, as shown in Figure 27.

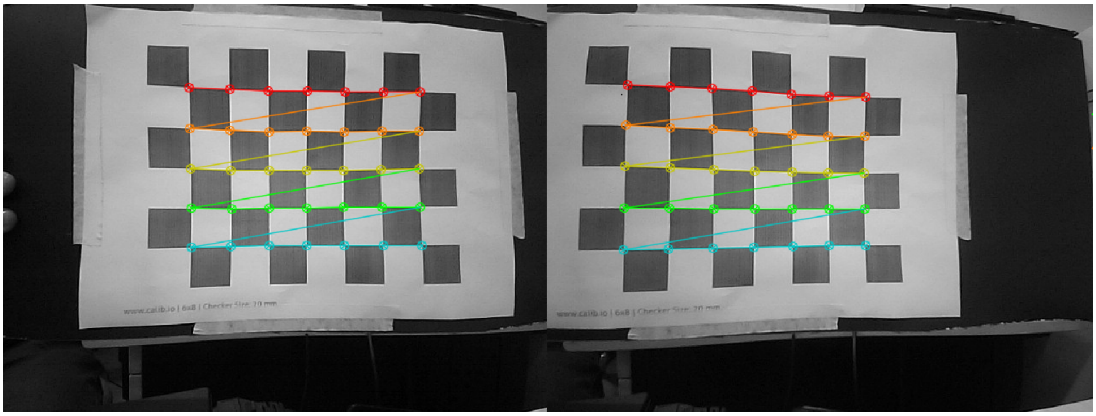

Figure 16: Calibration of a stereo camera with a 6x8 chessboard pattern

5.3.2 Top view camera

In the context of computer vision, planar homography is defined as a projective mapping from one plane to another. A direct example of planar homography is the mapping of a planar surface in space to the planar surface of the image. It is possible to express this relation in terms of matrix multiplication with use of homogeneous coordinates, each given point in a plane with coordinates $[x\ y\ 1]$ may be mapped to a new point in a different plane with coordinates $[u\ v\ 1]$ through the equation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Where the **H** is the 3x3 homography matrix. With a set of 4 points correspondence between the two planes it is possible to fully describe the homography relation. When more than 4 corresponding points are available,

5.3.3 Scene recognition

A system with 4 stereo cameras is proposed, one on each side of the car, similar as the one utilized by Tesla's cars (116). There are several options to use the data provided by this set of cameras, such as directly feed a Visual Simultaneous Localization and Mapping (VSLAM) algorithm in order to navigate in the environment. It is also possible

to take advantage of the known position of the cameras and infer the depth of the objects surrounding the car and get the similar data as a LiDAR sensor would provide. At last, it is even possible to directly feed a deep neural network (DNN) model directly to provide the position of the vehicle in relation to a known map.

The CARLA simulator was used to test the different approaches and an example of scene reconstruction is shown in Figures 17 and 18.

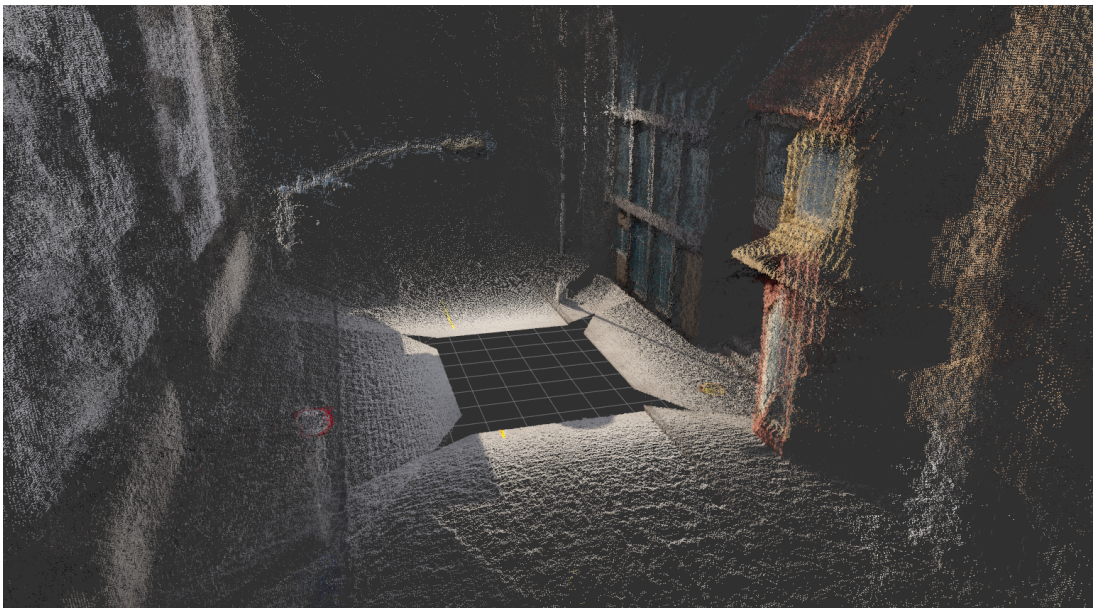

Figure 17: Point cloud inferred from 8 cameras in the Carla simulator

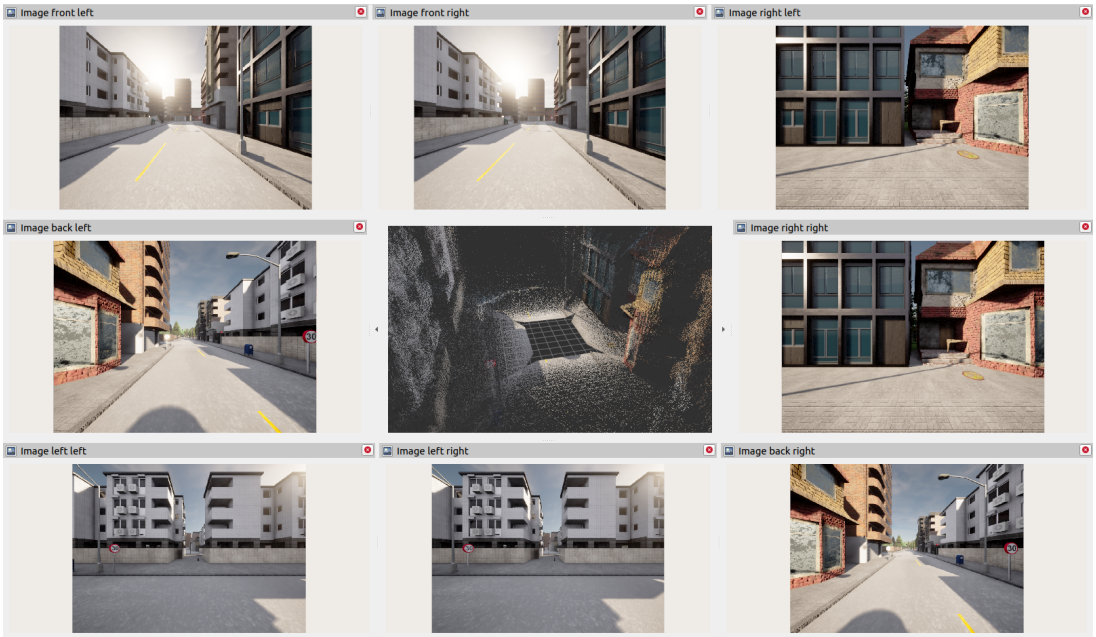

Figure 18: Detail with all 8 cameras in the virtual car

The images generated by the simulation are matched to infer the depth of each point in the image by computing its horizontal displacement in each image. The geometrical

principle is illustrated in Figure 20. To speed up computations, the image processing and matching is done via GPU processing, in such a way the system is capable of performing real-time computation.

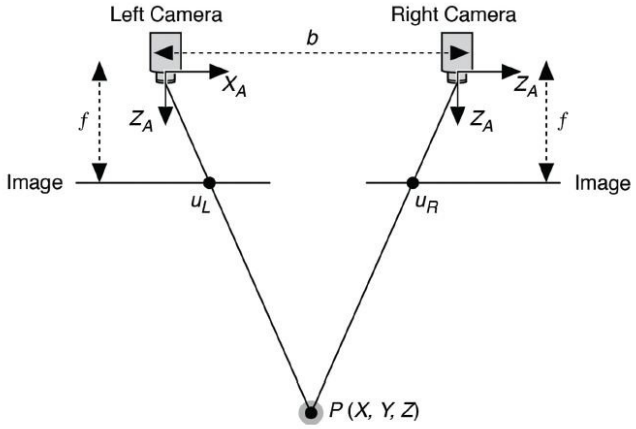

Figure 19: Geometrical principle of depth inference from stereo cameras, as illustrated by (10). The baseline is denoted by b and the focal distance is denoted by f

5.3.4 YOLO

The images obtained from the proposed camera system can be used to feed a deep neural network for object detection. As an autonomous vehicle relies on fast response times to function properly, a YOLO based solution was chosen. Its most recent official version is currently YOLOv7, which outperforms its previous versions in lower parameter density, faster inference times and higher precision (11).

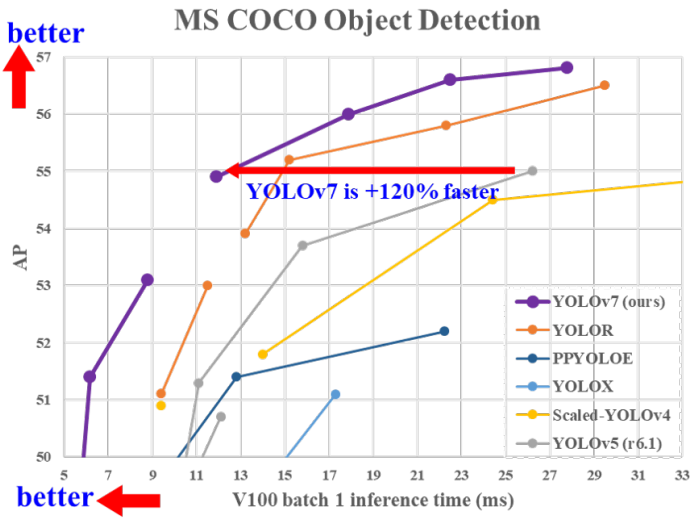

Figure 20: Inference time and AP comparison between YOLOv7 and other previous versions (11)

5.3.4.1 Structure

YOLOv7 has a few released variations, some aim for to improve its precision, while others focus on improving its performance. The relevant one to this project is the YOLOv7-tiny, which reduces the network size, number of parameters and changes its activation function to LeakyReLU, reducing significantly image processing time, at the cost of lower overall accuracy.

For comparison, both versions were trained for 50 epochs and tested in a validation computer, using a NVidia GTX 1050 and a 480 x 640 image as the network input. Overall, YOLOv7-tiny processed the images approximately x4 faster than its counter part, however with a slightly lower precision.

Therefore, even though YOLOv7 displays better performance, as the detection network will need to process four images at a time, the processing time gain is more valuable for a real time perception system.

Besides that, there was also an attempt to use the YOLOv7-tiny in a different framework than Pytorch: NCNN. It is a high-performance neural network inference framework made in C++ and optimized for the mobile and edge platforms, such as the Jetson Xavier NX used in the hardware prototype. To use it, the weights file must be converted from a .pt file to .onnx and then to both a .bin and .param files. However, when testing it on the Jetson Xavier board, it performed similarly to the original Pytorch version. So, as there was an increased maintenance cost due to the added conversions and no gain in performance, this attempt was put aside.

5.3.4.2 Dataset

The dataset used to train the neural network is a combination of FSOCO dataset, which consists of images manually labeled by the Formula Student Driverless community (117), and an authored dataset, built with pictures containing cones, taken around USP main campus, and labeled manually using the LabelImg software (118). An example of each dataset is displayed in Figures 21 and 22.To join both datasets, the Roboflow site was used [1].

The cones in the FSOCO dataset are grouped into 5 distinct classes: 'yellow cone', 'blue cone', 'orange cone', 'large orange cone' and 'unknown cone', as illustrated in Figure 23, with the two former being the vast majority of it. To contour the problem of over

[1]The site is available at the address https://roboflow.com/

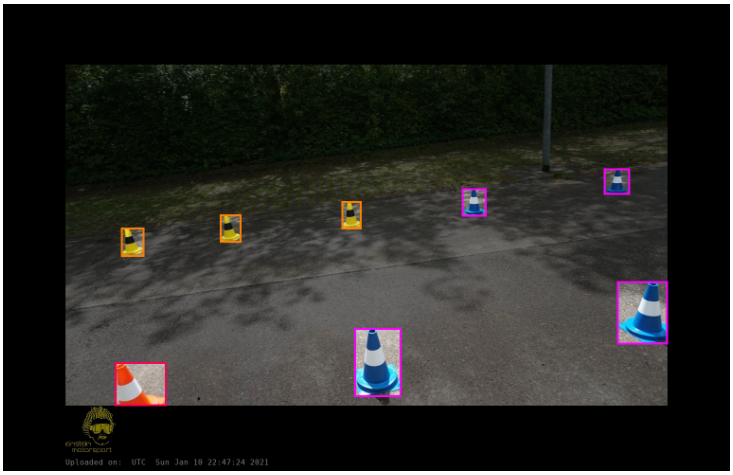

Figure 21: Example of an image from the FSOCO dataset

Figure 22: Example of an image from the autoral dataset

and under representation, displayed by the Figure 23, all objects where unified into the 'cone' label. This is not prejudicial in this project as the information to distinct between cone types is not relevant. The cones are being used as a generic marker or obstacle, so it is only necessary to know if it's a cone. Additionally, the 'unknown cone' label was removed, as it mostly contained small cones, which lost resolution when the images were resized for training.

Finally, before exporting the dataset, some techniques of data augmentation were applied to it, as it has shown to improve the overall precision of deep learning methods, while also reducing the overfit present in smaller datasets (119). An example of an image with data augmentation is displayed in Figure 24 and the characteristics of the resulting dataset are displayed in Figure 25.

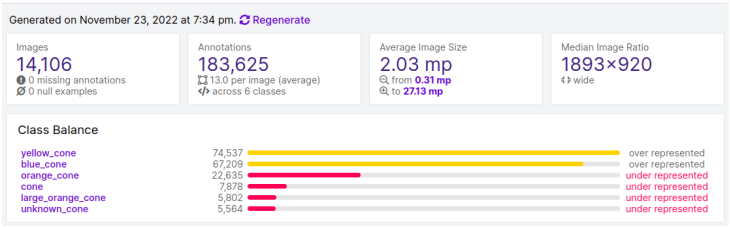

Figure 23: Distributions of classes in the cone dataset, generated by Roboflow's healthcheck feature

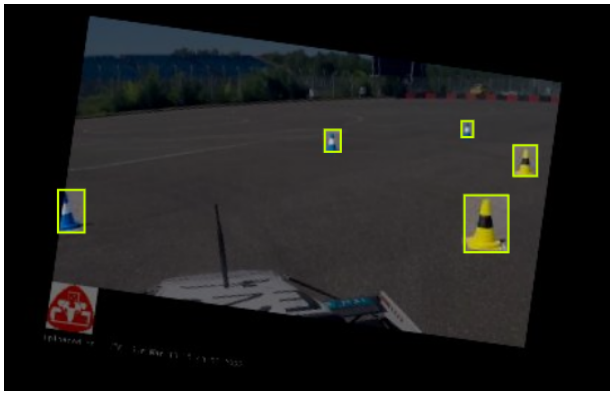

Figure 24: Example of data augmentation result, using brightness and rotation variation

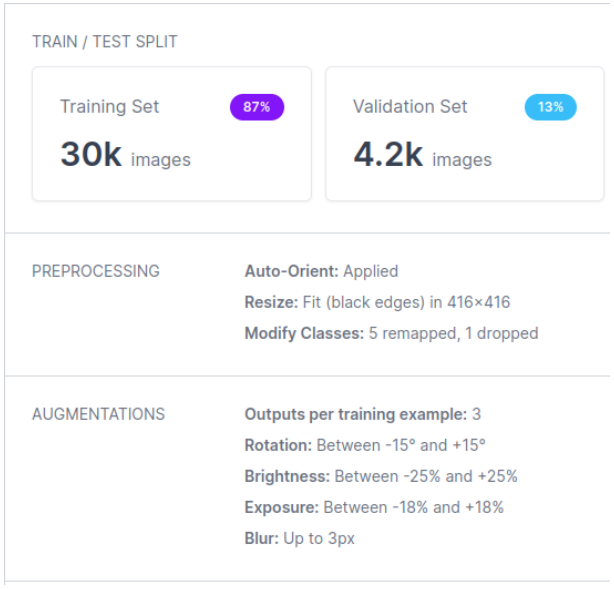

Figure 25: Final cone dataset characteristics

5.4 Control

5.4.1 Sliding modes control

5.4.1.1 Lateral control

In order to follow the desired path generated by the higher level routine, a sliding mode controller was implemented. The implemented control law is a modification from

the proposed in (120):

$$\delta = -\gamma \ \text{sign}\,(\psi - \psi_d) + v\dot{\psi}_d \tag{5.1}$$

Where

$$\gamma = v\left(\frac{a_{11}}{v}|\beta| + \frac{a_{12}}{v^2}|\dot{\psi}| + |\beta|a_0\frac{|\dot{v}|}{|v|}\right) + \frac{\gamma_0}{2} \tag{5.2}$$

δ is the control action (signal sent to the steering actuator), ψ is the heading angle of the vehicle, ψ_d is the desired heading angle, v is the velocity of the vehicle, γ_0 is an adjust parameter, a_{11} and a_{12} are constants dependent on the vehicle geometry, β is the vehicle sideship angle.

To prevent chattering, the function $\text{sign}\,(\psi - \psi_d)$ is changed to $\text{sat}\left(\frac{\psi-\psi_d}{\Phi}\right)$, where Φ defines a boundary layer around the sliding mode. Thus, the final form of the control law is

$$\delta = -\gamma \ \text{sat}\left(\frac{\psi - \psi_d}{\Phi}\right) + v\dot{\psi}_d \tag{5.3}$$

5.4.1.2 Longitudinal control

The longitudinal control of the vehicle is performed by a PID controller, whose desired velocity is described by the equation

$$v_d = \gamma_p\rho + \lambda \tag{5.4}$$

where γ_p and λ are adjust parameters and ρ is the distance to the followed point

6 RESULTS

6.1 Perception

To properly select the YOLOv7 structure, the average detection time for a 640x480 image was analysed, whose results are shown in Table 1.

	YOLOv7	YOLOv7-tiny (Pytorch)	YOLOv7-tiny (NCNN)
GeForce GTX 1050Ti	83 ms	21 ms	-
Jetson Xavier NX	-	36 ms	52 ms

Table 1: YOLOv7 average detection time with different hardware

The cone detection network was trained in YOLOv7-tiny, up to 500 epochs. Resulting in weights with more than 90% prediction, as displayed in Table 2.

Prediction	Recall	mAP@.5	mAP@.95
0.904	0.533	0.579	0.331

Table 2: YOLOv7-tiny performance metrics

This training resulted in a very sturdy cone detection system. However, false positives are also present, with detection confidences up to 45%. This can be resolved by increasing the detection confidence threshold.

In order to assist the driver while performing maneuvers with the car, an assistance system was developed in order to provide a third person view from above of the vehicle. This system was achieved using a homography transform as described in 5.3.2.

Four points A, B, C, D were chosen forming a square in the floor and their coordinates in the image plane were determined using the projection matrix of each camera, derived

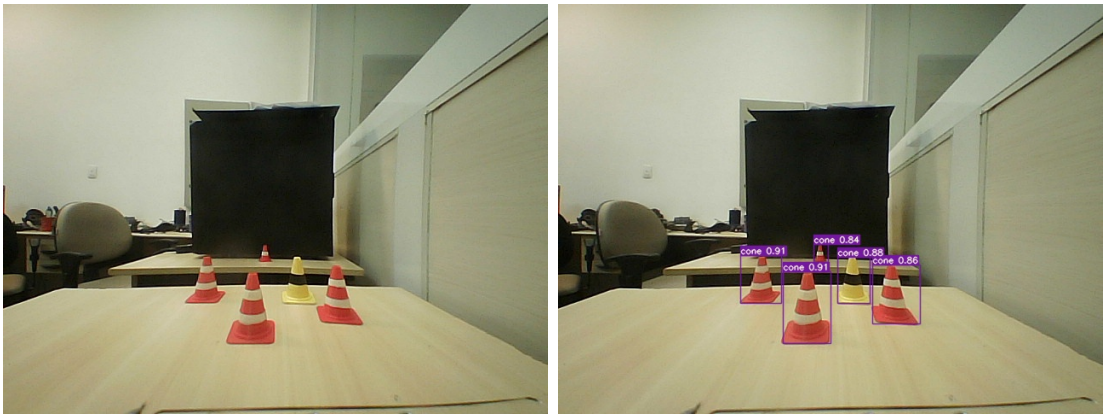

Figure 26: Results of the cone detection network

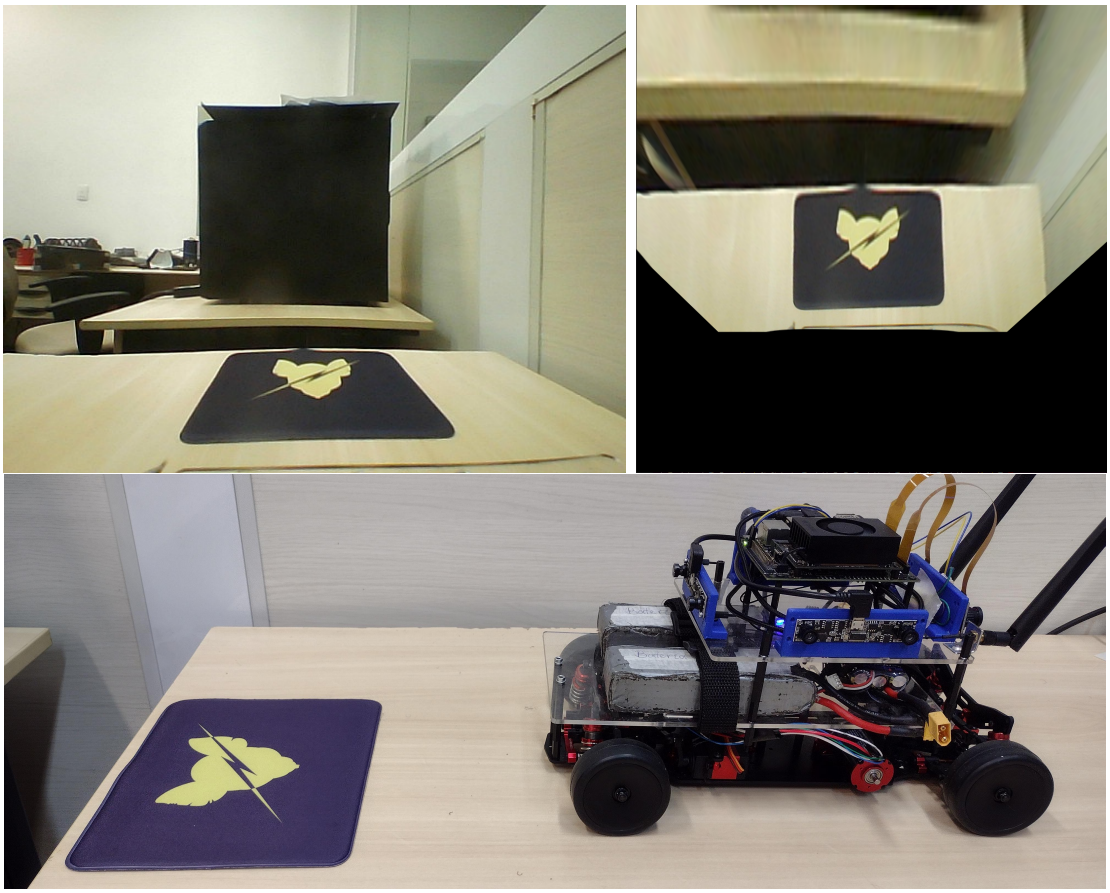

Figure 27: Top view virtual camera assistance with a homography transform

in the calibration process. Those points define the frame of the top view image. Finally, the homography matrix is numerically found with the Levenberg–Marquardt optimization algorithm (121, 122).

6.1.1 Localization

Once the detection of the landmarks is made, the perception system localizes the traffic cones in space in relation to itself. Transformation is made based in two assumptions: the surroundings of the vehicle are flat and the landmarks have a known height. With both assumptions, the Y coordinate in the camera frame of the center of all landmarks is equal to half of its height. It is then possible to use this information to simplify the projection matrix of the camera and find the X and Z coordinates in the camera frame using the (x, y) coordinates of the center of the cone in the image frame.

In other words, the perspective projection performs the following transformation between (X, Y, Z) camera frame coordinates and (x, y) image frame coordinates.

$$x = -f\frac{X}{Z} \quad y = -f\frac{Y}{Z}$$

where f is the focal distance of the camera, measured in pixels. As the Y coordinate of the traffic cones is known, it is possible to find X and Z as follows:

$$X = Y\frac{x}{y} \quad Z = -Y\frac{f}{y}$$

The position of the cone in respect to the camera is then transformed to the external fixed tf2 referential /odom as defined by REP 105 (123). Given the assumption that the cones are static, the vehicle performs its localization in the world while moving in respect to the location of the cones. The result of the localization system may be seen in Figure 28 and 29.

In addition, the vehicle is capable of localizing itself even if the cones are temporary hidden, thanks to the estimation of its position, based on its velocity and steer angle, using an odometry system.

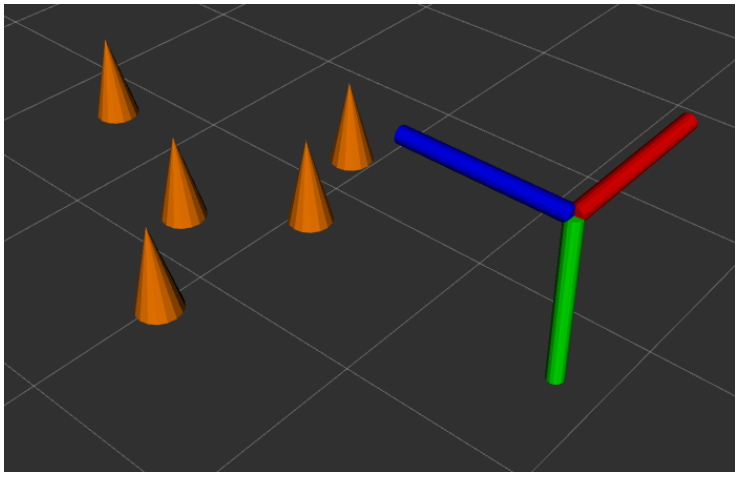

Figure 28: Graphical representation of the inferred position of the cones in respect to the camera reference frame

Figure 29: Photo of the real position of the detected cones

6.2 Control

6.2.1 Sliding modes control

The tests were performed in a fixed route inside CARLA map Town02. The path is followed from point A to point B, as shown in Figure 30. It contains 3 curves to the left, 3 curves to the right and 7 straight lines.

6.2.1.1 System performance

The validation was performed with a look ahead distance L of $2.0m$ and the variable estimations $\hat{a}_{11} = 2$, $\hat{a}_{12} = 0.2$, $\hat{a}_0 = 1$, $\Phi = 0.4$ and $\gamma_0 = 2$, defined using the car geometric and physic true parameters, as shown in Figure 31. The six disturbances correspond to each one of the curves displayed in the red path of the Figure 30, with the first 3 to the

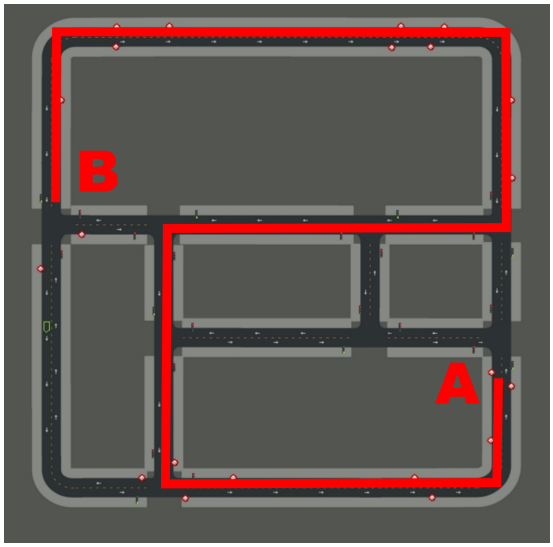

Figure 30: Illustration of CARLA map Town02 as provided by (12)

right, thus a positive value for the control action δ, and the latter 3 to the left, thus a negative value for the control action δ.

For the control of longitudinal speed, the parameter *lambda* was set to zero and γ_p was chosen in such a way that the maximum speed of the vehicle would be equal to $5m/s$ $(18km/h)$.

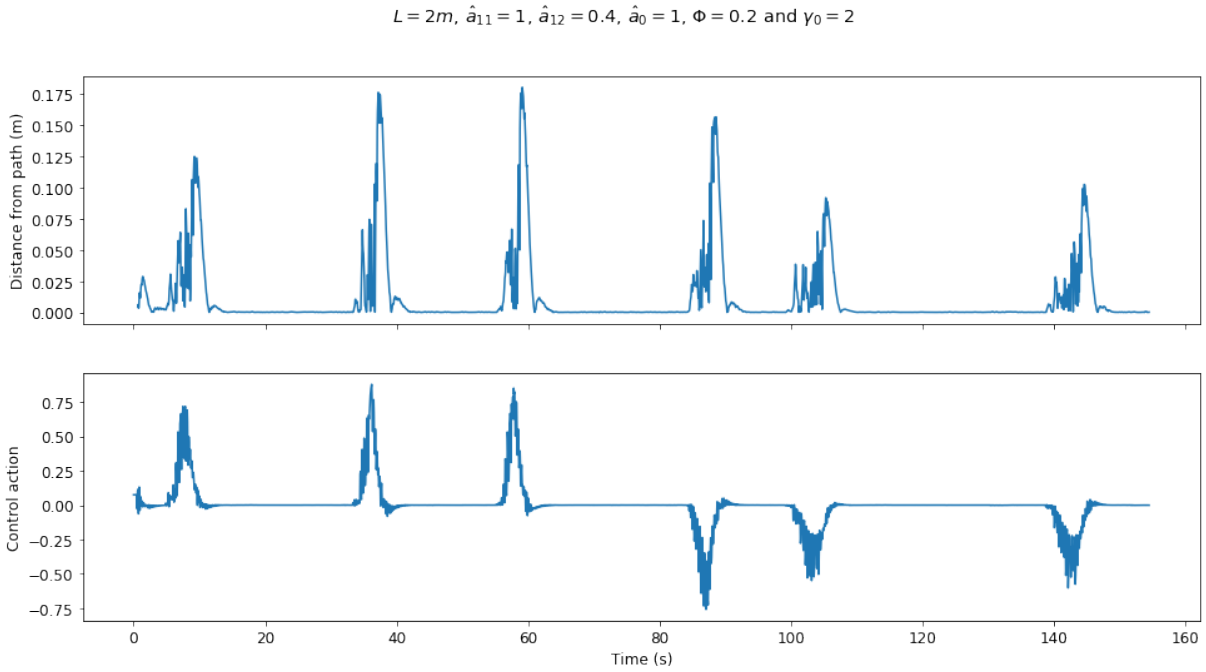

Figure 31: System performance for $L = 2.0m$, $\hat{a}_{11} = 2$, $\hat{a}_{12} = 0.2$, $\hat{a}_0 = 1$, $\Phi = 0.4$ and $\gamma_0 = 2$

6.2.1.2 Look ahead distance

The optimal look ahead distance was found to be $L = 2.0m$. A comparison of $L = 1.0m$, $L = 2.0m$ and $L = 5.0m$ is shown in Figure 32.

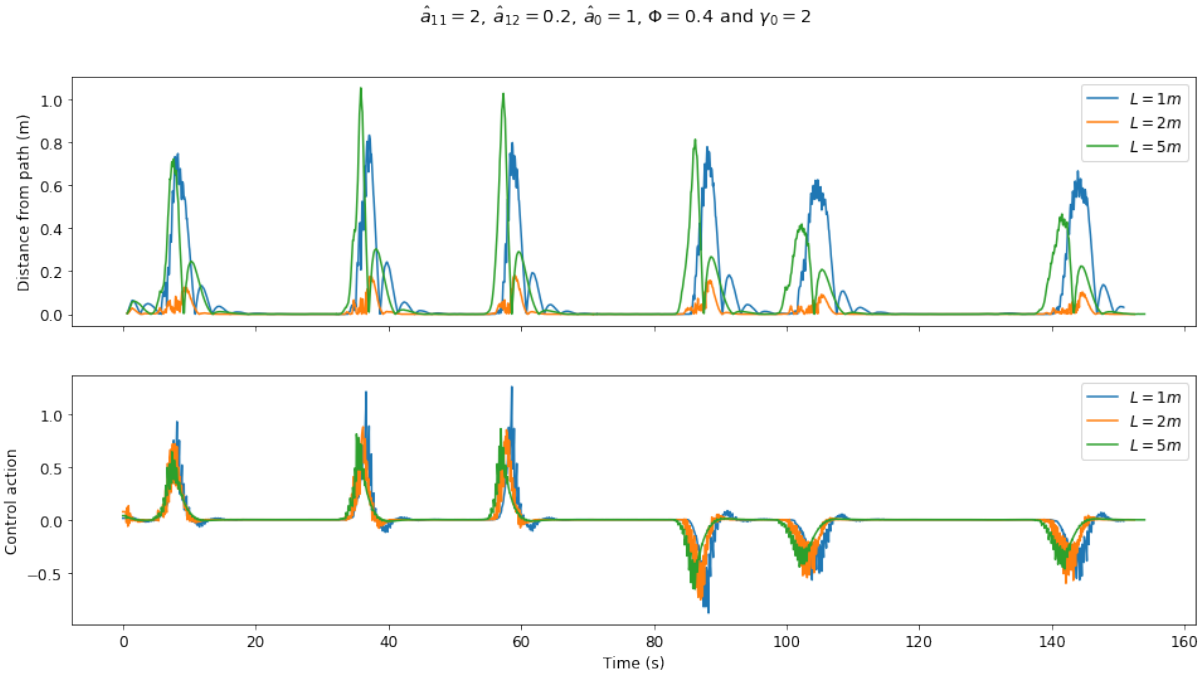

Figure 32: Comparison of different look ahead distances

6.2.1.3 Boundary layer

The effect of the boundary layer can be seen in Figure 33, setting $\Phi = 0.2$ and then $\Phi = 0.4$. It is possible to see that the boundary layer successfully prevents chattering without significantly compromise the performance of the system.

6.3 Discussion

The developed system was able to perceive the surrounding environment and detect the standardized landmarks in order to localize itself. The position estimate of the landmarks was performed using previous knowledge about its dimensions and the intrinsic parameters of the cameras in order to perform a inverse perspective projection. It was not possible to use the desired stereo configuration due to the challenge of setting all calibration parameters of the stereo matching algorithm, whose poor results made it impossible to use in the project. A possible fix may be the use of a fourth order radial distortion model for the cameras calibration instead of the third order used in this work.

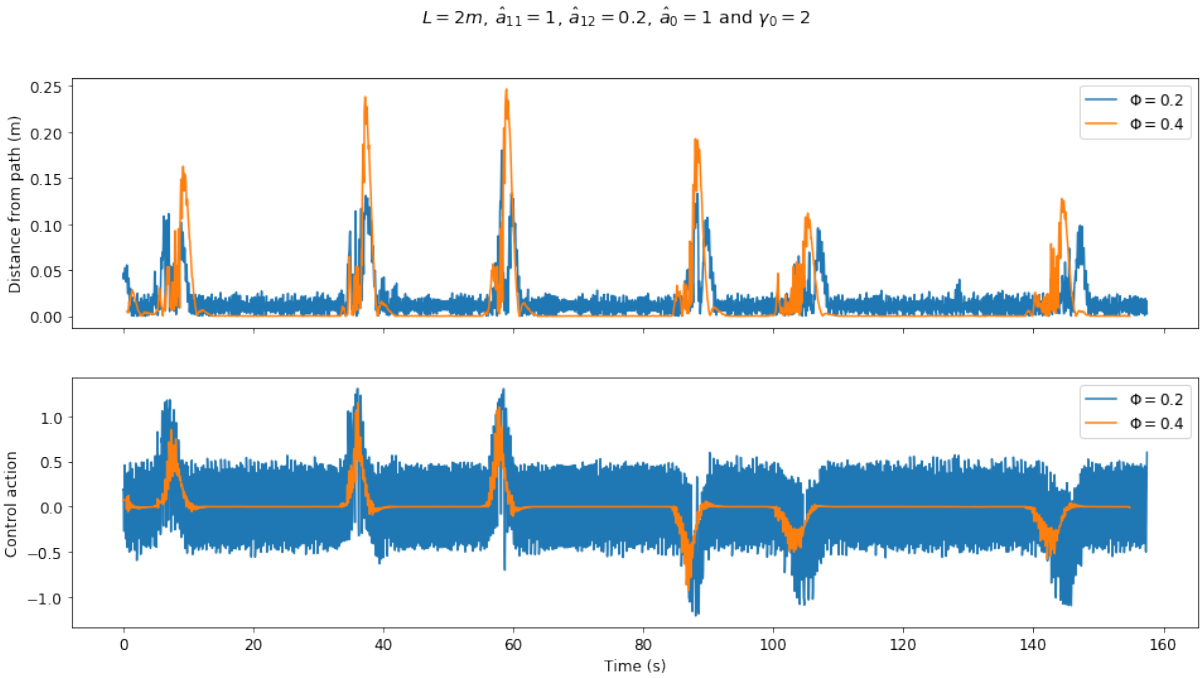

Figure 33: Illustration of the effect of a boundary layer with $\Phi = 0.2$ and $\Phi = 0.4$

In addition to the perception system, the top view camera also provides assistance to the driver when performing maneuvers in constrained environments. The proper calibration of all cameras was key in order to achieve good performance of this tool. The landmark detection using YOLOv7 proof to be robust and suitable to the task of object recognition in embedded devices. Its tiny version offered a good balance between performance and computational cost.

The implemented control algorithm was able to stabilize the system and guide a virtual vehicle through a desired pattern. Due to time constrains, it was not possible to validate it in the physical model, only in the simulated one. A possible following work for this project would be porting the implemented controller in the physical model.

The use of the modular framework ROS 2 makes the developed code for this work loosely coupled and easily extensible, in such a way that it is possible to robust existing libraries run alongside our code, such as the ROS Navigation Stack 2.

7 FINAL REMARKS

7.1 Conclusion

Road transportation plays a major role in our society at the same time that imposes a series of issues to human life, from pollution to traffic accidents. Car crashes alone correspond to the largest cause of death of young people from 5 to 29 years (17) and despite the complexity involved in the working principles of a automobile, human error is still the major cause of car accidents (19). In this context, the research and development of advanced driving assisting systems (ADS) arises as a possible solution to this problem.

This work presented a review of the state-of-the-art in the field of autonomous vehicles. A simulation model was prepared for initial testing and validation, then a physical prototype was built in order to test the system in real world conditions. The system was capable of detecting miniaturized traffic cones and localizing itself in respect to the traffic cones. A control system was also developed and tested in the simulation environment.

The system was built using the modular framework ROS 2, that enabled the development of modular code and the use of consolidated open source libraries from the field of robotics

7.2 Going further

The work done in this project may be expanded with further use of the built physical prototype. The use of robust planning systems, such as ROS Navigation Stack 2(64) or Autoware.AI may extend the capabilities of the platform into performing more complex tasks. Our project was unable to take advantage of the stereo configuration of the camera, so the depth measurement was restricted to objects with known geometry. The proper calibration of the stereo fusion algorithm may provide a more robust depth inferring system and so improve the systems performance.

The addition of different sensors, such as a inertial measurement unit (IMU) and a Lidar may provide a deeper understanding of the surroundings of the model and improve its perception. Different control techniques such as model predictive control (MPC), dynamic window approach (DWA) (94) and time elastic band (TEB) (124) could be tested in order to find the one that delivers the best performance.

7.3 Open source contributions

The presented work was built upon several free and open source projects and some improvements could be made to some of them during the development period. By the time of writing, seven different projects received contributions in the form of pull requests in GitHub in the context of this project. It is a small return given to the community in recognition of the meaningful work and assistance provided.

REFERENCES

1

2 SAE International, "Sae J3016TM levels of driving automation." [Online]. Available: ⟨https://www.sae.org/binaries/content/assets/cm/content/blog/sae-j3016-visual-chart_5.3.21.pdf⟩

3 D. Hood and W. Woodall, "ROS 2 update," in *RosCon 2016 - Seoul*, Oct. 2016. [Online]. Available: ⟨https://roscon.ros.org/2016/presentations/ROSCon2016-ROS2Update.pdf⟩

4 DDS Foundation, "What is DDS?" ⟨https://www.dds-foundation.org/what-is-dds-3/⟩, 09 2022.

5 S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017.

6 Autoware Foundation. (2021) Autonomous valet parking demonstration. Accessed: 2022-06-29. [Online]. Available: ⟨https://web.archive.org/web/20210731035452/https://autowarefoundation.gitlab.io/autoware.auto/AutowareAuto/avpdemo.html⟩

7 K. L. Talvala, K. Kritayakirana, and J. C. Gerdes, "Pushing the limits: From lanekeeping to autonomous racing," *Annual Reviews in Control*, vol. 35, no. 1, pp. 137–148, 2011.

8 J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *arXiv preprint arXiv:2202.07008*, 2022.

9 S. Tariq. (2021) AV development at massive scale. [Online]. Available: ⟨https://www.youtube.com/watch?v=3WqooCcUpn0⟩

10 M. Häringer, P. Schneider, S. Krusch, M. Dölzer, and U. Göhner, "Skeleton tracking in 3d space with two microsoft kinect," 02 2015.

11 C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

12 A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

13 H. Link, C. Nash, A. Ricci, and J. Shires, "A generalized approach for measuring the marginal social costs of road transport in europe," *International Journal of Sustainable Transportation*, vol. 10, no. 2, pp. 105–119, 2016.

14 S. A. Bagloee, M. Tavana, M. Asadi, and T. Oliver, "Autonomous vehicles: challenges, opportunities, and future implications for transportation policies," *Journal of modern transportation*, vol. 24, no. 4, pp. 284–303, 2016.

15 I. W. Parry, M. Walls, and W. Harrington, "Automobile externalities and policies," *Journal of economic literature*, vol. 45, no. 2, pp. 373–399, 2007.

16 J. I. Levy, J. J. Buonocore, and K. Von Stackelberg, "Evaluation of the public health impacts of traffic congestion: a health risk assessment," *Environmental health*, vol. 9, no. 1, pp. 1–12, 2010.

17 World Health Organization, *Global status report on road safety 2018*. Geneva: World Health Organization, 2018.

18 ——, "Global Health Estimates 2020: Deaths by Cause, Age, Sex, by Country and by Region, 2000-2019," Geneva, 2020. [Online]. Available: ⟨{https://www.who.int/data/gho/data/themes/mortality-and-global-health-estimates/ghe-leading-causes-of-death}⟩

19 S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," Washington, DC, Tech. Rep., 2015. [Online]. Available: ⟨{https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115}⟩

20 R. E. Stern, S. Cui, M. L. Delle Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, H. Pohlmann, F. Wu, B. Piccoli *et al.*, "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 205–221, 2018.

21 J. Lee and K. M. Kockelman, "Energy implications of self-driving vehicles," in *98th Annual Meeting of the Transportation Research Board in Washington, DC*, 2019.

22 P. Stenquist, "The motor car of the future. oakl. trib," 1918.

23 L. Clemmons, C. Jones, J. Dunn, and Kimball, "Vehicles capable of dynamic vision." [Online]. Available: ⟨https://www.youtube.com/watch?v=L3funFSRAbU⟩

24 E. D. Dickmanns and V. Graefe, "Dynamic monocular machine vision," *Machine vision and applications*, vol. 1, no. 4, pp. 223–240, 1988.

25 E. D. Dickmanns *et al.*, "Vehicles capable of dynamic vision," in *IJCAI*, vol. 97, 1997, pp. 1577–1592.

26 B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

27 Carnegie Mellon University NAVLAB, "No hands across america journal." [Online]. Available: ⟨http://www.cs.cmu.edu/afs/cs/usr/tjochem/www/nhaa/Journal.html⟩

28 S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the DARPA grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

29 C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

30 M. Buehler, K. Iagnemma, and S. Singh, *The 2005 DARPA grand challenge: the great robot race*. Springer, 2007, vol. 36.

31 ——, *The DARPA urban challenge: autonomous vehicles in city traffic*. springer, 2009, vol. 56.

32 A. Faisal, M. Kamruzzaman, T. Yigitcanlar, and G. Currie, "Understanding autonomous vehicles," *Journal of transport and land use*, vol. 12, no. 1, pp. 45–72, 2019.

33 "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," Society Automotive Engineers International (SAE), Standard, Apr. 2021. [Online]. Available: ⟨https://www.sae.org/standards/content/j3016_202104⟩

34 J. Wang, J. Liu, and N. Kato, "Networking and communications in autonomous driving: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1243–1274, 2018.

35 M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

36 G. Pardo-Castellote, "OMG data-distribution service: Architectural overview," in *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.* IEEE, 2003, pp. 200–206.

37 S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

38 Open Robotics, "About ROS 2 client libraries," ⟨https://docs.ros.org/en/rolling/Concepts/About-ROS-2-Client-Libraries.html⟩.

39 D. Thomas, "ROS 2 middleware interface," ⟨https://design.ros2.org/articles/ros_middleware_interface.html⟩, 09 2017.

40 J. Kay, "Proposal for implementation of real-time systems in ROS 2," ⟨https://design.ros2.org/articles/realtime_proposal.html⟩, 01 2016.

41 OMG DDS Security, "Dds security," Object Management Group, Geneva, CH, Standard, 07 2018. [Online]. Available: ⟨https://www.omg.org/spec/DDS-SECURITY/1.1⟩

42 K. Fazzari, "ROS 2 DDS-security integration," ⟨https://design.ros2.org/articles/ros2_dds_security.html⟩, 07 2020.

43 Open Robotics, "ROS 2 humble hawksbill release!" ⟨https://www.openrobotics.org/blog/2022/5/24/ros-2-humble-hawksbill-release⟩, 05 2022.

44 ——, "About different ROS 2 DDS/RTPS vendors," ⟨https://docs.ros.org/en/humble/Concepts/About-Different-Middleware-Vendors.html⟩, 2022.

45 M. Arguedas, S. Ragnarok, D. Thomas, and A. Nash, "ROS 2 releases and target platforms," Open Robotics, Standard REP 2000, 11 2021. [Online]. Available: ⟨https://ros.org/reps/rep-2000.html⟩

46 A. IST, "Opensplice announcement," ⟨https://github.com/ADLINK-IST/opensplice/blob/a4818af5f5931834501055cd7ce2339de4f0af13/README.md⟩, 06 2021.

47 OSRF, "ROS 2 default RMW TSC reports," ⟨https://osrf.github.io/TSC-RMW-Reports/⟩, 2022.

48 K. Scott, W. Woodall, and C. Lalancette, "2020 ROS Middleware Evaluation Report," ⟨https://osrf.github.io/TSC-RMW-Reports/galactic/⟩, 11 2020.

49 Open Robotics, "ROS 2 galactic geochelone release," ⟨https://www.openrobotics.org/blog/2021/5/24/ros-2-galactic-geochelone-release⟩, 05 2021.

50 K. Scott, C. Lalancette, and A. Nash, "2021 ROS Middleware Evaluation Report," ⟨https://osrf.github.io/TSC-RMW-Reports/humble/⟩, 10 2021.

51 K. Scott, "ROS 2 TSC Meeting Minutes for 2022-07-21," ⟨https://discourse.ros.org/t/ros-2-tsc-meeting-minutes-07-21-2022/26647⟩, 07 2022.

52 Y. Maruyama, S. Kato, and T. Azumi, "Exploring the performance of ROS 2," in *Proceedings of the 13th International Conference on Embedded Software*, 2016, pp. 1–10.

53 A. Dabrowski, R. Kozik, and M. Maciaś, "Evaluating the resilience of ROS 2 communication layer," 2016.

54 A. Pemmaiah, D. Pangercic, D. Aggarwal, K. Neumann, and K. Marcey, "Performance testing in ROS 2," ⟨https://www.apex.ai/post/performance-testing-in-ros-2⟩, 2018.

55 T. Kronauer, J. Pohlmann, M. Matthé, T. Smejkal, and G. Fettweis, "Latency analysis of ROS 2 multi-node systems," in *2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2021, pp. 1–7.

56 C. Bédard, I. Lütkebohle, and M. Dagenais, "ros2_tracing: Multipurpose low-overhead framework for real-time tracing of ROS 2," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6511–6518, 2022.

57 J. Fernandez, B. Allen, P. Thulasiraman, and B. Bingham, "Performance study of the robot operating system 2 with QoS and cyber security settings," in *2020 IEEE International Systems Conference (SysCon)*. IEEE, 2020, pp. 1–6.

58 L. Puck, P. Keller, T. Schnell, C. Plasberg, A. Tanev, G. Heppner, A. Roennau, and R. Dillmann, "Performance evaluation of real-time ROS 2 robotic control in a time-synchronized distributed network," 08 2021, pp. 1670–1676.

59 A. Sadat, S. Casas, M. Ren, X. Wu, P. Dhawan, and R. Urtasun, "Perceive, predict, and plan: Safe motion planning through interpretable semantic representations," in *European Conference on Computer Vision*. Springer, 2020, pp. 414–430.

60 C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixao, F. Mutz *et al.*, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113816, 2021.

61 S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, "Autoware on board: Enabling autonomous vehicles with embedded systems," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 287–296.

62 B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed., ser. Advanced Textbooks in Control and Signal Processing. Springer, 2009.

63 E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 300–307.

64 S. Macenski, F. Martín, R. White, and J. G. Clavero, "The marathon 2: A navigation system," *CoRR*, vol. abs/2003.00368, 2020. [Online]. Available: ⟨https://arxiv.org/abs/2003.00368⟩

65 R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

66 A. Merzlyakov and S. Macenski, "A comparison of modern general-purpose visual slam approaches," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

67 N. Le Large, F. Bieder, and M. Lauer, "Comparison of different slam approaches for a driverless race car," *tm-Technisches Messen*, vol. 88, no. 4, pp. 227–236, 2021.

68 S. Srinivasan, I. Sa, A. Zyner, V. Reijgwart, M. I. Valls, and R. Siegwart, "End-to-end velocity estimation for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6869–6875, 2020.

69 C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

70 S. Sumikura, M. Shibuya, and K. Sakurada, "OpenVSLAM: A versatile visual SLAM framework," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 2292–2295.

71 M. Labbé and F. Michaud, "RTAB-map as an open-source LIDAR and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

72 W. Liu, S. Liao, W. Ren, W. Hu, and Y. Yu, "High-level semantic feature detection: A new perspective for pedestrian detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5187–5196.

73 L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, "Lidar–camera fusion for road detection using fully convolutional neural networks," *Robotics and Autonomous Systems*, vol. 111, pp. 125–131, 2019.

74 J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

75 J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

76 ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

77 A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

78 E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

79 P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

80 P. Thombre, "Multi-objective path finding using reinforcement learning," 2018.

81 G. Khekare, P. Verma, U. Dhanre, S. Raut, and S. Sheikh, "The optimal path finding algorithm based on reinforcement learning," *International Journal of Software Science and Computational Intelligence (IJSSCI)*, vol. 12, no. 4, pp. 1–18, 2020.

82 J. Yu, Y. Su, and Y. Liao, "The path planning of mobile robot by neural networks and hierarchical reinforcement learning," *Frontiers in Neurorobotics*, p. 63, 2020.

83 H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck, "Route planning in transportation networks," in *Algorithm engineering*. Springer, 2016, pp. 19–80.

84 A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson *et al.*, "Odin: Team victortango's entry in the DARPA urban challenge," *Journal of field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.

85 J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. Lee, A. Stewart, P. Vernaza, J. Derenick, J. Spletzer, and B. Satterfield, "Little ben: The ben franklin racing team's entry in the 2007 DARPA urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 598–614, 2008.

86 C. R. Baker and J. M. Dolan, "Traffic interaction in the urban challenge: Putting boss on its best behavior," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 1752–1758.

87 M. Colledanchise and P. Ögren, *Behavior trees in robotics and AI: An introduction*. CRC Press, 2018.

88 J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

89 S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

90 S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

91 L. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration for fast path planning," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE, 1994, pp. 2138–2145.

92 L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

93 J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

94 D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

95 C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*. VDE, 2012, pp. 1–6.

96 J. Ni and J. Hu, "Path following control for autonomous formula racecar: Autonomous formula student competition," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1835–1840.

97 D. Ha and J. Schmidhuber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.

98 A. Brunnbauer, L. Berducci, A. Brandstätter, M. Lechner, R. Hasani, D. Rus, and R. Grosu, "Model-based versus model-free deep reinforcement learning for autonomous racing cars," *arXiv preprint arXiv:2103.04909*, 2021.

99 A. Liniger, X. Zhang, P. Aeschbach, A. Georghiou, and J. Lygeros, "Racing miniature cars: Enhancing performance using stochastic mpc and disturbance feedback," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 5642–5647.

100 U. Rosolia and F. Borrelli, "Learning how to autonomously race a car: a predictive control approach," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2713–2719, 2019.

101 D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

102 "Road vehicles - Functional safety - Part 1: Vocabulary," International Organization for Standardization, Geneva, Standard, 2018. [Online]. Available: ⟨https://www.iso.org/obp/ui/#iso:std:iso:26262:-1:ed-2:v1:en⟩

103 K. Conley, T. Straszheim, M. Kjaergaard, E. Rublee, T. Foote, B. Gerkey, J. O'Quin, and D. Thomas, "ROS Package Naming," Open Robotics, Standard REP 144, 12 2021. [Online]. Available: ⟨https://ros.org/reps/rep-0144.html⟩

104 The Robotics Back-End. Package organization for a ROS stack [best practices]. Accessed: 2022-06-29. [Online]. Available: ⟨https://roboticsbackend.com/package-organization-for-a-ros-stack-best-practices⟩

105 E. Gamma, R. Helm, R. Johnson, R. E. Johnson, J. Vlissides *et al.*, *Design patterns: elements of reusable object-oriented software*. Pearson Deutschland GmbH, 1995.

106 "Segurança no trâfego – Cones para sinalização viária," Associação Brasileira de Normas Técnicas, Rio de Janeiro, Standard, 2020.

107 K. B. C. Carliss Y. Baldwin, *Design Rules, Vol. 1: The Power of Modularity*. The MIT Press, 2000.

108 TESLA, "Model 3." [Online]. Available: ⟨https://www.tesla.com/model3⟩

109 CARLA, "ROS/ROS 2 bridge for carla simulator," ⟨https://github.com/carla-simulator/ros-bridge⟩, 2022.

110 *EC-4pole 30*, Maxom Group, 3 2021, ⟨https://www.maxongroup.com/medias/sys_master/root/8882556829726/EN-21-259.pdf⟩.

111 Benjamin Vedder, "The VESC project," ⟨https://vesc-project.com/⟩, 2017.

112 C. B. Duane, "Close-range camera calibration," *Photogramm. Eng*, vol. 37, no. 8, pp. 855–866, 1971.

113 J. G. Fryer and D. C. Brown, "Lens distortion for close-range photogrammetry," *Photogrammetric engineering and remote sensing*, vol. 52, pp. 51–58, 1986.

114 A. Kaehler and G. Bradski, *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. " O'Reilly Media, Inc.", 2016.

115 D. C. Brown, "Decentering distortion of lenses," *Photogrammetric Engineering and Remote Sensing*, 1966.

116 A. Karpathy. (2021) Tesla CVPR 2021 workshop on autonomous driving. [Online]. Available: ⟨https://www.youtube.com/watch?v=g6bOwQdCJrc⟩

117 N. Vödisch, D. Dodel, and M. Schötz, "Fsoco: The formula student objects in context dataset," *SAE International Journal of Connected and Automated Vehicles*, vol. 5, no. 12-05-01-0003, 2022.

118 Tzutalin, "Labelimg," ⟨https://github.com/tzutalin/labelImg⟩, 2015.

119 C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

120 A. Stotsky and X. Hu, "Sliding mode control of a car-like mobile robot using single-track dynamic model," *Variable structure systems, sliding mode and nonlinear control*, pp. 181–191, 1999.

121 D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

122 K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

123 W. Meeussen, "Coordinate Frames for Mobile Platforms," Open Robotics, Standard REP 144, 10 2010. [Online]. Available: ⟨https://www.ros.org/reps/rep-0105.html⟩

124 C. Rösmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control," in *2015 European Control Conference (ECC)*, 2015, pp. 3352–3357.